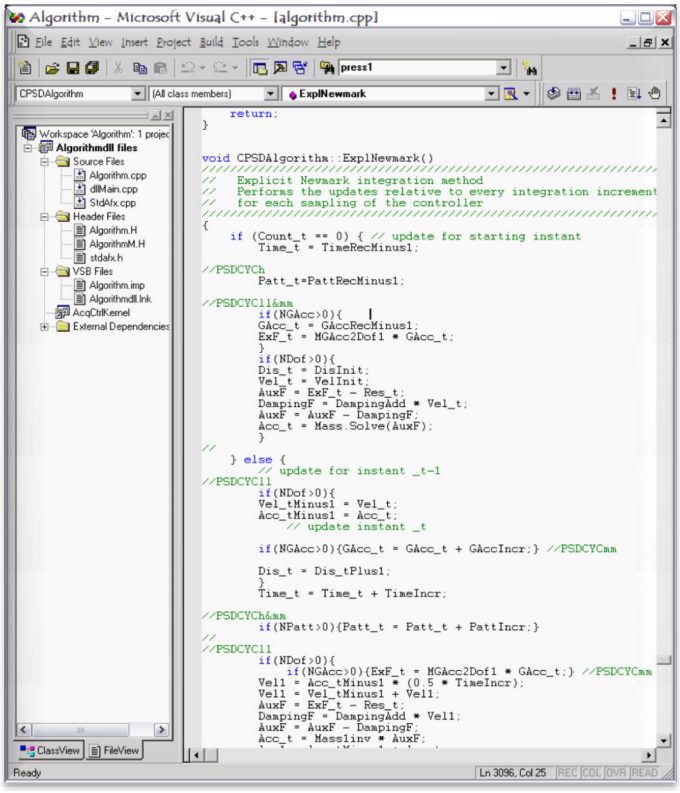
JRC Scientific and Technical Reports

Annex

Master configuration files

- Beatriz Zapico Blanco, F. Javier Molina



EUR 23448 EN/2 - 2008

JRC EUROPEAN COMMISSION

ipsc Institute for the Protection and Security of the Citizen

The Institute for the Protection and Security of the Citizen provides research-based, systems-oriented support to EU policies so as to protect the citizen against economic and technological risk. The Institute maintains and develops its expertise and networks in information, communication, space and engineering technologies in support of its mission. The strong cross-fertilisation between its nuclear and non-nuclear activities strengthens the expertise it can bring to the benefit of customers in both domains.

European Commission
Joint Research Centre
Institute for the Protection and Security of the Citizen

Contact information
Address: ELSA Laboratory, IPSC, Joint Research Centre, via Enrico Fermi 2749, 21027 Ispra, Italy
E-mail: beatriz.zapico@jrc.it
Tel.: 0332785712
Fax: 0332785379

http://ipsc.jrc.ec.europa.eu/
http://www.jrc.ec.europa.eu/

JRC 47201

EUR 23448 EN/2
ISBN 978-92-79-09706-5
ISSN 1018-5593
DOI 10.2788/92806

Luxembourg: Office for Official Publications of the European Communities

Printed in Italy

EXAMPLE j01

Example1 1 pattern 1 slave controller

A. TESTNAME.TXT

```
j01
****The first line of this file contains the name of the
**** test to be executed by the dll algorithm. It should contain four
**** characters or less.
```

B. Data input file j01_dat.txt

```
>>>Data of the test
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line
>>>
>>>----------------------General Data-------------------------------
>>>
>TEST NAME: j01
>TITLE DESCRIBING THE TEST:
Example1 1 pattern 1 slave controller

>NUMBER OF SLAVE CONTROLLERS CONSIDERED AT THIS MASTER NCon>0:
      1

>EXTERNAL BOARD NUMBER OF EVERY SLAVE CONTROLLER CtrNum (1,NCon):
      1

>PISTON SECTION1 (TENSION CHAMBER) AT Con1 Con2 ...  Section1 (1,NCon)
kN/Bar:
      2.75

>PISTON SECTION2 (COMPRESSION CHAMBER) AT Con1 Con2 ...  Section2 (1,NCon)
kN/Bar:
```

```
      2.75

>NUMBER OF PATTERN INPUT FILES NPatt>=0:
      1

>NUMBER OF PSD DEGREES OF FREEDOM NDof>=0:
      0

>IF NDof>0, NUMBER OF GROUND ACCELERATION INPUT FILES NGAcc>=0:

>IF NDof>0, NUMBER OF STRAIN-RATE DEPENDENT DEVICES TO BE COMPENSATED AT
THE RESTORING FORCES NSR>=0:

>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS AT THE PATTERN AND GROUND
ACELERATION INPUT FILES TimeRecIncr s:
      0.02

>NUMBER OF INTERNAL CONTROLLER 2ms SAMPLINGS BETWEEN TWO RECORDS InterRec:
      1000

>>>Time scale lambda = InterRec*0.002/TimeRecIncr = RealTime/PrototypeTime

>PROTOTYPE TIME FOR NEXT TEST STOP TimeStop s:
      30.0

>>>
>>>-------------------------Pattern Data----------------------------
>>>

>>>Formula for computation of every pattern 1<= M <=NPatt:

>>>IntM(Rec)=IntM(Rec-1) + PattISpanM/100 * PattFileM(Rec-1) * TimeRecIncr

>>>PattM(Rec) = PattSpanM/100 * PattFileM(Rec) + IntM(Rec)

>IF NPatt>0, PROPORTIONAL SPAN PERCENTAGE MULTIPILIER PattSpan (1,NPatt)
%:
      50

>IF NPatt>0, INTEGRAL SPAN PERCENTAGE MULTIPLIER PattISpan (1,NPatt) %/s:
      0

>IF NPatt>0, FILE NAME FOR EVERY PATTERN (UP TO 4 CHARACTERS PER NAME)
PattName (NPatt, 4):
      unif

>IF NPatt>0, INFLUENCE MATRIX FROM PATTERNS TO TARGETS Patt2Targ
(NCon,NPatt) (mm OR kN)/(mm OR kN):
      1000

>>>
>>>-------------------Other Influence Matrices----------------------
>>>
```

```
>INFLUENCE MATRIX FROM HEIDENHAIN TO TARGETS Heid2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0

>INFLUENCE MATRIX FROM TEMPOSONICS TO TARGETS Temp2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0

>INFLUENCE MATRIX FROM LOAD CELLS TO TARGETS LCell2Targ (NCon,NCon) (mm OR
kN)/kN:
      0.0

>INFLUENCE MATRIX FROM FORCE2 CHANNEL TO TARGETS Force22Targ (NCon,NCon)
(mm OR kN)/unit:
      0.0

>INFLUENCE MATRIX FROM SPEED CHANNEL TO TARGETS Speed2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0

>INFLUENCE MATRIX FROM LVDT CHANNEL TO TARGETS Lvdt2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0

>>>
>>>----------------------PsD equation data---------------------------
>>>

>IF NDof>0, THEORETICAL MASS MATRIX Mass(NDof,NDof) kg:

>IF NDof>0, THEORETICAL ADDITIONAL STIFFNESS MATRIX StiffAdd(NDof,NDof)
N/m:

>IF NDof>0, THEORETICAL ADDITIONAL DAMPING MATRIX DampingAdd(NDof,NDof)
Ns/m:

>IF NDof>0, INITIAL DISPLACEMENT DisInit(1,NDof) m:

>IF NDof>0, INITIAL VELOCITY VelInit(1,NDof) m/s:

>>>To avoid restoring-force offset compensation introduce NFSampl=0

>IF NDof>0, NUMBER OF SAMPLINGS TO AVERAGE FOR RESTORING FORCE OFFSET
COMPUTATION NFSampl:

>IF NDof>0 AND NFSampl>0, PRESCRIBED RESTORING FORCE VALUE FOR OFFSET
COMPUTATION ResInit(1,NDof) N:

>IF NGacc>0, PROPORTIONAL SPAN PERCENTAGE MULTIPLIER GAccSpan(1,NGAcc) %:

>>>Accelerogram time increment must be equal to prototype time increment

>IF NGAcc>0, FILE NAME FOR EVERY GROUND ACCELEROGRAM (UP TO 4 CHARACTERS
PER NAME) GAccName(NGAcc,4):
```

```
>IF NDof>0 AND NGAcc>0, INFLUENCE MATRIX FROM GROUND MOTION TO DoF
GAcc2Dof(NDof,NGAcc) (m/s/s)/(m/s/s):

>IF NDof>0, INFLUENCE MATRIX FROM LOAD CELLS TO RESTORING FORCES
LCell2Res(NDof,NCon) N/kN:

>IF NDof>0, INFLUENCE MATRIX FROM DoF DISPLACEMENTS TO TARGETS
Dis2Targ(NCon,NDof) mm/m:

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO FSR
Force22FSR(NSR,NCon) kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO FSR Speed2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO FSR Lvdt2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO DSR
Force22DSR(NSR,NCon) mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO DSR Speed2DSR(NSR,NCon)
mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO DSR Lvdt2DSR(NSR,NCon)
mm/unit:

>>>Formula for strain-rate-compensation additional force at every device:
>>>FSRAdd = SRFacF0*FSR + SRFacD0*DSR + SRFacF1*FSRdot + SRFacD1*DSRdot

>IF NSR>0, FORCE CORRECTION FACTOR SRFacF0(1,NSR) kN/kN:

>IF NSR>0,  DISPLACEMENT CORRECTION FACTOR SRFacD0(1,NSR) kN/mm:

>IF NSR>0,  FORCE-DERIVATIVE CORRECTION FACTOR SRFacF1(1,NSR) kNs/kN:

>IF NSR>0,  DISPLACEMENT-DERIVATIVE CORRECTION FACTOR SRFacD1(1,NSR)
kNs/mm:

>IF NSR>0, INFLUENCE MATRIX FROM FSRAdd TO RESTORING FORCES
FSR2Res(NDof,NSR) N/kN:

>>>
>>>----------------------Algo Alarm Data-----------------------------
>>>

>ALGO_ALARM SUPERIOR LIMIT AT HEIDENHAIN HeidMax (1,NCon)
      10
>ALGO_ALARM INFERIOR LIMIT AT HEIDENHAIN HeidMin (1,NCon)
      -10
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS TempMax (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS TempMin (1,NCon)
```

```
      -1e10
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS ABS TempAbsMax (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS ABS TempAbsMin (1,NCon)
      -1e10
>ALGO_ALARM SUPERIOR LIMIT AT LOAD CELL FORCE LCellMax (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT LOAD CELL FORCE LCellMin (1,NCon)
      -1e10
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR ErrorMax (1,NCon)
      1e10
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR AVERAGE ErrAvMax (1,NCon)
      1e10
>ALGO_ALARM LIMIT AT ABSOLUTE ENERGY ERROR AVERAGE EneErAvMax
      1e10
>ALGO_ALARM SUPERIOR LIMIT AT LVDT LvdtMax (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT LVDT LvdtMin (1,NCon)
      -1e10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION1 Press1Max (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT PRESSION1 Press1Min (1,NCon)
      -1e10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION2 Press2Max (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT PRESSION2 Press2Min (1,NCon)
      -1e10
>ALGO_ALARM SUPERIOR LIMIT AT SERVOVALVE ServoMax (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT SERVOVALVE ServoMin (1,NCon)
      -1e10
```

C. Pattern History unif_pat.txt

```
>>>Pattern history
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line

>TITLE OF THE RECORD:
      Uniform 100 mm, 5000 points

>NUMBER OF RECORD POINTS OF THIS HISTORY  NRecPatt:
5000

>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS TimeRecIncr s:
      0.02

>RECORD VALUES Patt (NRecPatt,1) mm or kN:
 100
```

```
 100
 100
(…)
```

D. MASTER.BAT

```
####################################
## Menu.Bat Sample
####################################


#### TEMPOSONIC CHANNEL USED ######
####
#### USED     = 1
#### NOT USED = 0

set C1.UseTempo1=0
set C1.UseTempo2=1
set C2.UseTempo1=0
set C2.UseTempo2=1
set C3.UseTempo1=0
set C3.UseTempo2=1
set C4.UseTempo1=0
set C4.UseTempo2=1


#### TYPE OF TEMPOSONIC USED ######
####
#### PARALLEL (100 um) = 1      #OLD MODEL
#### SSI      (2 um)   = 2      #NEW MODEL

set C1.TypeTempo1=1
set C1.TypeTempo2=1
set C2.TypeTempo1=1
set C2.TypeTempo2=1
set C3.TypeTempo1=1
set C3.TypeTempo2=1
set C4.TypeTempo1=1
set C4.TypeTempo2=1


########################################
##   PID PARAMETERS
##
##

set C1.DispP=1.0
set C1.DispI=100
set C1.DispD=0
set C2.DispP=1.0
set C2.DispI=100
```

```
set C2.DispD=0
set C3.DispP=1.0
set C3.DispI=100
set C3.DispD=0
set C4.DispP=1.0
set C4.DispI=100
set C4.DispD=0

set C1.SafeDispP=0.3
set C1.SafeDispI=2000
set C1.SafeDispD=0
set C2.SafeDispP=0.3
set C2.SafeDispI=2000
set C2.SafeDispD=0
set C3.SafeDispP=0.3
set C3.SafeDispI=2000
set C3.SafeDispD=0
set C4.SafeDispP=0.3
set C4.SafeDispI=2000
set C4.SafeDispD=0

####################################
##      CONVERSION FACTORS
##
##

set MEASURE1-CF.Disp-H=1
set MEASURE1-CF.Disp-T1=1
set MEASURE1-CF.Disp-T2=-1
set MEASURE1-CF.Force1=100
set MEASURE1-CF.Force2=100
set MEASURE1-CF.Spool=1
set MEASURE1-CF.Acc=1
set MEASURE1-CF.Speed=1
set MEASURE1-CF.Disp-L=1
set MEASURE1-CF.Pressure1=80
set MEASURE1-CF.Pressure2=80
set MEASURE1-CF.AD9=1
set MEASURE1-CF.AD10=1
set MEASURE1-CF.AD11=1
set MEASURE1-CF.AD12=1
set MEASURE1-CF.AD13=1
set MEASURE1-CF.AD14=1
set MEASURE1-CF.AD15=1
set MEASURE1-CF.AD16=1

set OUTPUT1-CF.Servo1=-1

# For Tempo and Heide analog output on Dac1 and Dac2 10V/1000mm=0.01
set OUTPUT1-CF.Dac1=0.01
set OUTPUT1-CF.Dac2=0.01

set MEASURE2-CF.Disp-H=1
set MEASURE2-CF.Disp-T1=1
```

```
set MEASURE2-CF.Disp-T2=-1
set MEASURE2-CF.Force1=100
set MEASURE2-CF.Force2=100
set MEASURE2-CF.Spool=1
set MEASURE2-CF.Acc=1
set MEASURE2-CF.Speed=1
set MEASURE2-CF.Disp-L=1
set MEASURE2-CF.Pressure1=80
set MEASURE2-CF.Pressure2=80
set MEASURE2-CF.AD9=1
set MEASURE2-CF.AD10=1
set MEASURE2-CF.AD11=1
set MEASURE2-CF.AD12=1
set MEASURE2-CF.AD13=1
set MEASURE2-CF.AD14=1
set MEASURE2-CF.AD15=1
set MEASURE2-CF.AD16=1

set OUTPUT2-CF.Servo1=-1


# For conv factor of Heide3 minus sign was used to invert the measurement
set MEASURE3-CF.Disp-H=-1
set MEASURE3-CF.Disp-T1=1
set MEASURE3-CF.Disp-T2=-1
set MEASURE3-CF.Force1=100
set MEASURE3-CF.Force2=100
set MEASURE3-CF.Spool=1
set MEASURE3-CF.Acc=1
set MEASURE3-CF.Speed=1
set MEASURE3-CF.Disp-L=1
set MEASURE3-CF.Pressure1=80
set MEASURE3-CF.Pressure2=80
set MEASURE3-CF.AD9=1
set MEASURE3-CF.AD10=1
set MEASURE3-CF.AD11=1
set MEASURE3-CF.AD12=1
set MEASURE3-CF.AD13=1
set MEASURE3-CF.AD14=1
set MEASURE3-CF.AD15=1
set MEASURE3-CF.AD16=1

set OUTPUT3-CF.Servo1=-1


set MEASURE4-CF.Disp-H=1
set MEASURE4-CF.Disp-T1=1
set MEASURE4-CF.Disp-T2=-1
set MEASURE4-CF.Force1=100
set MEASURE4-CF.Force2=100
set MEASURE4-CF.Spool=1
set MEASURE4-CF.Acc=1
set MEASURE4-CF.Speed=1
set MEASURE4-CF.Disp-L=1
```

```
set MEASURE4-CF.Pressure1=80
set MEASURE4-CF.Pressure2=80
set MEASURE4-CF.AD9=1
set MEASURE4-CF.AD10=1
set MEASURE4-CF.AD11=1
set MEASURE4-CF.AD12=1
set MEASURE4-CF.AD13=1
set MEASURE4-CF.AD14=1
set MEASURE4-CF.AD15=1
set MEASURE4-CF.AD16=1

set OUTPUT4-CF.Servo1=-1


####################################
##  ANTI SPIKE VALUES
##
##

set ANTISPIKE1.AntiSpikeForce1=20
set ANTISPIKE1.AntiSpikeTempo2=1
set ANTISPIKE1.AntiSpikeHeide=1
set ANTISPIKE1.AntiSpikeLvdt=1

set ANTISPIKE2.AntiSpikeForce1=20
set ANTISPIKE2.AntiSpikeTempo2=1
set ANTISPIKE2.AntiSpikeHeide=1
set ANTISPIKE2.AntiSpikeLvdt=1

set ANTISPIKE3.AntiSpikeForce1=20
set ANTISPIKE3.AntiSpikeTempo2=1
set ANTISPIKE3.AntiSpikeHeide=1
set ANTISPIKE3.AntiSpikeLvdt=1

set ANTISPIKE4.AntiSpikeForce1=20
set ANTISPIKE4.AntiSpikeTempo2=1
set ANTISPIKE4.AntiSpikeHeide=1
set ANTISPIKE4.AntiSpikeLvdt=1


####################################
## ALARM LEVELS
##
##

set ALARM1.ErrorDelta=1
set ALARM1.Inserted=1
set ALARM2.ErrorDelta=1
set ALARM2.Inserted=1
set ALARM3.ErrorDelta=1
set ALARM3.Inserted=1
set ALARM4.ErrorDelta=1
set ALARM4.Inserted=1
```

```
####################################
##  MASTER ALGORITHM TYPE
##
##

#Set PSD.DllAlgorithm=PSD_TRAD.DLL
#Set PSD.DllAlgorithm=PSDCYCp.DLL
#Set PSD.DllAlgorithm=CYC02.DLL
#Set PSD.DllAlgorithm=CYCSTOD.DLL
#Set PSD.DllAlgorithm=PSD_SUB.DLL
#Set PSD.DllAlgorithm=PSD05.DLL
Set PSD.DllAlgorithm=PSDCYC03.DLL

####################################
##  PROCEDURE FOR SPECIAL WIRING
##

PROCEDURE INTERNALALGOOUTPUT1.DAC1=INTERNALALGOINPUT1.Tempo2
PROCEDURE INTERNALALGOOUTPUT1.DAC2=INTERNALALGOINPUT1.Heide

PROCEDURE START
```

E. MASTER.INI

```
####
#### MASTER.INI
#### 21/05/07
####
########################################

;*****************
; GENERAL INFO   *
;*****************

[GENERAL]
NAME=MASTER
TRACE=YES
NETWORK=YES
REMOTE=YES
WEB_REMOTE=YES
FTP_SERVER=YES
#### Specify the number of slave boards used in your servo-controller
NUM_SLAVE=4
#### Specify YES if you want to insert the Digital Output Debug (digital
outputs 9 to 16)
#### Default is NO
HWDEBUG=NO
# Length of memory block used in the interrupt
# to make the exchange with the master
# default = 8
INTERNALALGOINPUT_LENGTH1=16
```

```
INTERNALALGOOUTPUT_LENGTH1=16
INTERNALALGOINPUT_LENGTH2=16
INTERNALALGOOUTPUT_LENGTH2=16
INTERNALALGOINPUT_LENGTH3=16
INTERNALALGOOUTPUT_LENGTH3=16
INTERNALALGOINPUT_LENGTH4=16
INTERNALALGOOUTPUT_LENGTH4=16

########################################

;*****************
; NETWORK INFO   *
;*****************

[NETWORK]
#### NODE_NAME is the official name of the Master CPU inside the network
NODE_NAME=CTRL01
#### DEVICE_NAME=ETHER_0 means that you are using the Ethernet protocol
DEVICE_NAME=ETHER_0
#### DEVICE Type: can be I855 for Master board with Intel Network
Interfcace Controller integrated
####          or
####          NE2K for ISA board NE2000 with jumper correct setting
####               or RTL fot rtl 81xx NIC
DEVICE_TYPE=I855
FTP_SERVER=YES
#### WEB_SERVER=YES if telepresence is used
WEB_SERVER=NO

########################################

;*****************
; ETHER_0 Info   *
;*****************

[ETHER_0]
#### Insert the Master IP-ADDRESS =>
#### Internet:
IP_ADDRESS=192.168.0.186

#### Intranet:
# IP_ADDRESS=192.168.1.186
#### Subnet used in the Network
SUBNET_MASK=255.255.255.0
#### Gateway IP Address
#### Internet:
GW_ADDRESS=192.168.0.1
#### Intranet:
# GW_ADDRESS=192.168.1.1
#### Domain Name Server(DNS) IP Address
#### Internet:
DNS_ADDRESS=139.191.131.70
#### Intranet:
# DNS_ADDRESS=192.168.1.254
```

```
#### Specific setting for NE2000 ISA board
PORT=0x300
IRQ=5

########################################

;*****************
; TIMER Info     *
;*****************

[EXTERNAL_TIMER]
DEVICE_NAME=PCL_720_1
IRQ=10
SAMPLING_TIME=2
#[INTERNAL_TIMER]
#SAMPLING_TIME=1

;******************
; ACQCTRL_DEVICES *
;******************

[ACQCTRL_DEVICES]
NUMBER=1
DEV_1=PCL_720_1

;******************
;* PCL_720
;******************

[PCL_720_1]
TYPE=PCL_720
PORT=0x2A0

########################################

;*****************
; Trace Info     *
;*****************

[TRACE]

FILE=TRACE.LOG
NUM_LINE=100
DEVICE=ALL
MODE=ALL

########################################

;*****************
; Remote Info    *
;*****************

[REMOTE]
#### DEVICE_TYPE=DUALRAM_PARVUS
```

```
#### DEVICE_NAME=DUALRAM_PARVUS_1
DEVICE_TYPE=RT_SOCKET_SERVER
DEVICE_NAME=RT_SOCKET_1
[RT_SOCKET_1]
PORT=10000


;*****************
; Remote Info    *
;*****************
[WEB_REMOTE]
DEVICE_NAME=RT_SOCKET_2

[RT_SOCKET_2]
PORT=10080


########################################

;********************
; Disk Service Info *
;********************

[DISK_SERVICE]
PATH=C:\MASTER\
CACHE_SIZE=0XA0000
TEXT_MODE=FALSE
OVERWRITING_FILE=FALSE
```

F. HOST.CFG

```
|xxx.xxx.xxx.xxx|
|141. 63. 54.100|
|141. 63. 54.101|
|139.191.131.  *|
|139.191.254.128|
|192.168.  0.  *|
```

G. USERS.CFG

```
|login name| password |type|
|elsadat   |acq,98    |1   |
|super     |          |1   |
|privi     |          |2   |
|normal    |          |3   |
|guest     |          |4   |
```

EXAMPLE j02

Example2 1 accelerogram 1 DoF 1 slave controller

A. Data input file j02_dat.txt

```
>>>Data of the test
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line
>>>
>>>----------------------General Data-------------------------------
>>>
>TEST NAME: j02
>TITLE DESCRIBING THE TEST:
Example2 1 accelerogram 1 DoF 1 slave controller

>NUMBER OF SLAVE CONTROLLERS CONSIDERED AT THIS MASTER NCon>0:
      1

>EXTERNAL BOARD NUMBER OF EVERY SLAVE CONTROLLER CtrNum (1,NCon):
      1

>PISTON SECTION1 (TENSION CHAMBER) AT Con1 Con2 ...  Section1 (1,NCon)
kN/Bar:
      2.75

>PISTON SECTION2 (COMPRESSION CHAMBER) AT Con1 Con2 ...  Section2 (1,NCon)
kN/Bar:
      2.75

>NUMBER OF PATTERN INPUT FILES NPatt>=0:
      0

>NUMBER OF PSD DEGREES OF FREEDOM NDof>=0:
      1

>IF NDof>0, NUMBER OF GROUND ACCELERATION INPUT FILES NGAcc>=0:
```

```
      1

>IF NDof>0, NUMBER OF STRAIN-RATE DEPENDENT DEVICES TO BE COMPENSATED AT
THE RESTORING FORCES NSR>=0:
      0

>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS AT THE PATTERN AND GROUND
ACELERATION INPUT FILES TimeRecIncr s:
      0.02

>NUMBER OF INTERNAL CONTROLLER 2ms SAMPLINGS BETWEEN TWO RECORDS InterRec:
      1000

>>>Time scale lambda = InterRec*0.002/TimeRecIncr = RealTime/PrototypeTime

>PROTOTYPE TIME FOR NEXT TEST STOP TimeStop s:
      30.0

>>>
>>>-------------------------Pattern Data----------------------------
>>>

>>>Formula for computation of every pattern 1<= M <=NPatt:

>>>IntM(Rec)=IntM(Rec-1) + PattISpanM/100 * PattFileM(Rec-1) * TimeRecIncr

>>>PattM(Rec) = PattSpanM/100 * PattFileM(Rec) + IntM(Rec)

>IF NPatt>0, PROPORTIONAL SPAN PERCENTAGE MULTIPILIER PattSpan (1,NPatt)
%:

>IF NPatt>0, INTEGRAL SPAN PERCENTAGE MULTIPLIER PattISpan (1,NPatt) %/s:

>IF NPatt>0, FILE NAME FOR EVERY PATTERN (UP TO 4 CHARACTERS PER NAME)
PattName (NPatt, 4):

>IF NPatt>0, INFLUENCE MATRIX FROM PATTERNS TO TARGETS Patt2Targ
(NCon,NPatt) (mm OR kN)/(mm OR kN):


>>>
>>>-------------------Other Influence Matrices----------------------
>>>

>INFLUENCE MATRIX FROM HEIDENHAIN TO TARGETS Heid2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0

>INFLUENCE MATRIX FROM TEMPOSONICS TO TARGETS Temp2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0

>INFLUENCE MATRIX FROM LOAD CELLS TO TARGETS LCell2Targ (NCon,NCon) (mm OR
kN)/kN:
```

```
      0.0

>INFLUENCE MATRIX FROM FORCE2 CHANNEL TO TARGETS Force22Targ (NCon,NCon)
(mm OR kN)/unit:
      0.0

>INFLUENCE MATRIX FROM SPEED CHANNEL TO TARGETS Speed2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0

>INFLUENCE MATRIX FROM LVDT CHANNEL TO TARGETS Lvdt2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0

>>>
>>>----------------------PsD equation data---------------------------
>>>

>IF NDof>0, THEORETICAL MASS MATRIX Mass(NDof,NDof) kg:
      8300.0

>IF NDof>0, THEORETICAL ADDITIONAL STIFFNESS MATRIX StiffAdd(NDof,NDof)
N/m:
      0.0

>IF NDof>0, THEORETICAL ADDITIONAL DAMPING MATRIX DampingAdd(NDof,NDof)
Ns/m:
      0.0

>IF NDof>0, INITIAL DISPLACEMENT DisInit(1,NDof) m:
      0.0

>IF NDof>0, INITIAL VELOCITY VelInit(1,NDof) m/s:
      0.0

>>>To avoid restoring-force offset compensation introduce NFSampl=0

>IF NDof>0, NUMBER OF SAMPLINGS TO AVERAGE FOR RESTORING FORCE OFFSET
COMPUTATION NFSampl:
      5000

>IF NDof>0 AND NFSampl>0, PRESCRIBED RESTORING FORCE VALUE FOR OFFSET
COMPUTATION ResInit(1,NDof) N:
      0.0

>IF NGacc>0, PROPORTIONAL SPAN PERCENTAGE MULTIPLIER GAccSpan(1,NGAcc) %:

>>>Accelerogram time increment must be equal to prototype time increment

>IF NGAcc>0, FILE NAME FOR EVERY GROUND ACCELEROGRAM (UP TO 4 CHARACTERS
PER NAME) GAccName(NGAcc,4):
      cent
```

```
>IF NDof>0 AND NGAcc>0, INFLUENCE MATRIX FROM GROUND MOTION TO DoF
GAcc2Dof(NDof,NGAcc) (m/s/s)/(m/s/s):
      1.0

>IF NDof>0, INFLUENCE MATRIX FROM LOAD CELLS TO RESTORING FORCES
LCell2Res(NDof,NCon) N/kN:
      1000.0

>IF NDof>0, INFLUENCE MATRIX FROM DoF DISPLACEMENTS TO TARGETS
Dis2Targ(NCon,NDof) mm/m:
      1000.0

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO FSR
Force22FSR(NSR,NCon) kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO FSR Speed2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO FSR Lvdt2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO DSR
Force22DSR(NSR,NCon) mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO DSR Speed2DSR(NSR,NCon)
mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO DSR Lvdt2DSR(NSR,NCon)
mm/unit:

>>>Formula for strain-rate-compensation additional force at every device:
>>>FSRAdd = SRFacF0*FSR + SRFacD0*DSR + SRFacF1*FSRdot + SRFacD1*DSRdot

>IF NSR>0, FORCE CORRECTION FACTOR SRFacF0(1,NSR) kN/kN:

>IF NSR>0,  DISPLACEMENT CORRECTION FACTOR SRFacD0(1,NSR) kN/mm:

>IF NSR>0,  FORCE-DERIVATIVE CORRECTION FACTOR SRFacF1(1,NSR) kNs/kN:

>IF NSR>0,  DISPLACEMENT-DERIVATIVE CORRECTION FACTOR SRFacD1(1,NSR)
kNs/mm:

>IF NSR>0, INFLUENCE MATRIX FROM FSRAdd TO RESTORING FORCES
FSR2Res(NDof,NSR) N/kN:

>>>
>>>----------------------Algo Alarm Data-----------------------------
>>>
>ALGO_ALARM SUPERIOR LIMIT AT HEIDENHAIN HeidMax (1,NCon)
      10
>ALGO_ALARM INFERIOR LIMIT AT HEIDENHAIN HeidMin (1,NCon)
      -10
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS TempMax (1,NCon)
      1e10
```

```
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS TempMin (1,NCon)
      -1e10
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS ABS TempAbsMax (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS ABS TempAbsMin (1,NCon)
      -1e10
>ALGO_ALARM SUPERIOR LIMIT AT LOAD CELL FORCE LCellMax (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT LOAD CELL FORCE LCellMin (1,NCon)
      -1e10
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR ErrorMax (1,NCon)
      1e10
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR AVERAGE ErrAvMax (1,NCon)
      1e10
>ALGO_ALARM LIMIT AT ABSOLUTE ENERGY ERROR AVERAGE EneErAvMax
      1e10
>ALGO_ALARM SUPERIOR LIMIT AT LVDT LvdtMax (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT LVDT LvdtMin (1,NCon)
      -1e10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION1 Press1Max (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT PRESSION1 Press1Min (1,NCon)
      -1e10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION2 Press2Max (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT PRESSION2 Press2Min (1,NCon)
      -1e10
>ALGO_ALARM SUPERIOR LIMIT AT SERVOVALVE ServoMax (1,NCon)
      1e10
>ALGO_ALARM INFERIOR LIMIT AT SERVOVALVE ServoMin (1,NCon)
      -1e10
```

B. Ground Acceleration History octu_dat.txt

```
>>>Input ground acceleration history
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line
>TITLE OF THE RECORD:
   El Centro 3.4175 m/s/s, 20 s, NEFOREEE specified
>NUMBER OF RECORD POINTS OF THIS HISTORY  NRecGAcc:
1001
>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS TimeRecIncr s:
      0.02
>ACCELERATION VALUES GAcc (NRecGAcc,1) m/s/s:
0
-0.014002968
-0.10801309
-0.101011606
```

```
-0.088018656
  (…)
```

EXAMPLE N41

NEFOREEE 1DoF original frame. 57.4% El Centro

A. TESTNAME.TXT

```
n41
****The first line of this file contains the name of the
**** test to be executed by the dll algorithm. It should contain four
**** characters or less.
```

B. Data input file n41_dat.txt

```
>>>Data of the test
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line
>>>
>>>----------------------General Data-------------------------------
>>>
>TEST NAME: n41
>TITLE DESCRIBING THE TEST:
 NEFOREEE 1 DoF original frame. 57.4% El Centro

>NUMBER OF SLAVE CONTROLLERS CONSIDERED AT THIS MASTER NCon>0:
      2

>EXTERNAL BOARD NUMBER OF EVERY SLAVE CONTROLLER CtrNum (1,NCon):
      1     2

>PISTON SECTION1 (TENSION CHAMBER) AT Con1 Con2 ...  Section1 (1,NCon)
kN/Bar:
      2.75  2.75
```

```
>PISTON SECTION2 (COMPRESSION CHAMBER) AT Con1 Con2 ...  Section2 (1,NCon)
kN/Bar:
      2.75  2.75

>NUMBER OF PATTERN INPUT FILES NPatt>=0:
      0

>NUMBER OF PSD DEGREES OF FREEDOM NDof>=0:
      1

>IF NDof>0, NUMBER OF GROUND ACCELERATION INPUT FILES NGAcc>=0:
      1

>IF NDof>0, NUMBER OF STRAIN-RATE DEPENDENT DEVICES TO BE COMPENSATED AT
THE RESTORING FORCES NSR>=0:
      0

>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS AT THE PATTERN AND GROUND
ACELERATION INPUT FILES TimeRecIncr s:
      0.02

>NUMBER OF INTERNAL CONTROLLER 2ms SAMPLINGS BETWEEN TWO RECORDS InterRec:
      1000

>>>Time scale lambda = InterRec*0.002/TimeRecIncr = RealTime/PrototypeTime

>PROTOTYPE TIME FOR NEXT TEST STOP TimeStop s:
      30.0

>>>
>>>-------------------------Pattern Data----------------------------
>>>

>>>Formula for computation of every pattern 1<= M <=NPatt:

>>>IntM(Rec)=IntM(Rec-1) + PattISpanM/100 * PattFileM(Rec-1) * TimeRecIncr

>>>PattM(Rec) = PattSpanM/100 * PattFileM(Rec) + IntM(Rec)

>IF NPatt>0, PROPORTIONAL SPAN PERCENTAGE MULTIPILIER PattSpan (1,NPatt)
%:

>IF NPatt>0, INTEGRAL SPAN PERCENTAGE MULTIPLIER PattISpan (1,NPatt) %/s:

>IF NPatt>0, FILE NAME FOR EVERY PATTERN (UP TO 4 CHARACTERS PER NAME)
PattName (NPatt, 4):

>IF NPatt>0, INFLUENCE MATRIX FROM PATTERNS TO TARGETS Patt2Targ
(NCon,NPatt) (mm OR kN)/(mm OR kN):

>>>
>>>-------------------Other Influence Matrices----------------------
>>>
```

```
>INFLUENCE MATRIX FROM HEIDENHAIN TO TARGETS Heid2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM TEMPOSONICS TO TARGETS Temp2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM LOAD CELLS TO TARGETS LCell2Targ (NCon,NCon) (mm OR
kN)/kN:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM FORCE2 CHANNEL TO TARGETS Force22Targ (NCon,NCon)
(mm OR kN)/unit:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM SPEED CHANNEL TO TARGETS Speed2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM LVDT CHANNEL TO TARGETS Lvdt2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0   0.0
      0.0   0.0

>>>
>>>----------------------PsD equation data---------------------------
>>>

>IF NDof>0, THEORETICAL MASS MATRIX Mass(NDof,NDof) kg:
      8300.0

>IF NDof>0, THEORETICAL ADDITIONAL STIFFNESS MATRIX StiffAdd(NDof,NDof)
N/m:
      0.0

>IF NDof>0, THEORETICAL ADDITIONAL DAMPING MATRIX DampingAdd(NDof,NDof)
Ns/m:
      0.0

>IF NDof>0, INITIAL DISPLACEMENT DisInit(1,NDof) m:
      0.0

>IF NDof>0, INITIAL VELOCITY VelInit(1,NDof) m/s:
      0.0

>>>To avoid restoring-force offset compensation introduce NFSampl=0
```

```
>IF NDof>0, NUMBER OF SAMPLINGS TO AVERAGE FOR RESTORING FORCE OFFSET
COMPUTATION NFSampl:
      5000

>IF NDof>0 AND NFSampl>0, PRESCRIBED RESTORING FORCE VALUE FOR OFFSET
COMPUTATION ResInit(1,NDof) N:
      0.0

>IF NGacc>0, PROPORTIONAL SPAN PERCENTAGE MULTIPLIER GAccSpan(1,NGAcc) %:
      57.4

>>>Accelerogram time increment must be equal to prototype time increment
>IF NGAcc>0, FILE NAME FOR EVERY GROUND ACCELEROGRAM (UP TO 4 CHARACTERS
PER NAME) GAccName(NGAcc,4):
      cent

>IF NDof>0 AND NGAcc>0, INFLUENCE MATRIX FROM GROUND MOTION TO DoF
GAcc2Dof(NDof,NGAcc) (m/s/s)/(m/s/s):
      1.0

>IF NDof>0, INFLUENCE MATRIX FROM LOAD CELLS TO RESTORING FORCES
LCell2Res(NDof,NCon) N/kN:
      1000.0     1000.0

      >IF NDof>0, INFLUENCE MATRIX FROM DoF DISPLACEMENTS TO TARGETS
Dis2Targ(NCon,NDof) mm/m:
      1000.0
      1000.0

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO FSR
Force22FSR(NSR,NCon) kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO FSR Speed2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO FSR Lvdt2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO DSR
Force22DSR(NSR,NCon) mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO DSR Speed2DSR(NSR,NCon)
mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO DSR Lvdt2DSR(NSR,NCon)
mm/unit:

>>>Formula for strain-rate-compensation additional force at every device:
>>>FSRAdd = SRFacF0*FSR + SRFacD0*DSR + SRFacF1*FSRdot + SRFacD1*DSRdot

>IF NSR>0, FORCE CORRECTION FACTOR SRFacF0(1,NSR) kN/kN:

>IF NSR>0,  DISPLACEMENT CORRECTION FACTOR SRFacD0(1,NSR) kN/mm:
```

```
>IF NSR>0,  FORCE-DERIVATIVE CORRECTION FACTOR SRFacF1(1,NSR) kNs/kN:

>IF NSR>0,  DISPLACEMENT-DERIVATIVE CORRECTION FACTOR SRFacD1(1,NSR)
kNs/mm:

>IF NSR>0, INFLUENCE MATRIX FROM FSRAdd TO RESTORING FORCES
FSR2Res(NDof,NSR) N/kN:

>>>
>>>----------------------Algo Alarm Data-----------------------------
>>>

>ALGO_ALARM SUPERIOR LIMIT AT HEIDENHAIN HeidMax (1,NCon)
      50    50
>ALGO_ALARM INFERIOR LIMIT AT HEIDENHAIN HeidMin (1,NCon)
      -50   -50
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS TempMax (1,NCon)
      50    50
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS TempMin (1,NCon)
      -50   -50
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS ABS TempAbsMax (1,NCon)
      200   200
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS ABS TempAbsMin (1,NCon)
      100   100
>ALGO_ALARM SUPERIOR LIMIT AT LOAD CELL FORCE LCellMax (1,NCon)
      50    50
>ALGO_ALARM INFERIOR LIMIT AT LOAD CELL FORCE LCellMin (1,NCon)
      -50   -50
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR ErrorMax (1,NCon)
      0.4   0.4
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR AVERAGE ErrAvMax (1,NCon)
      0.1   0.1
>ALGO_ALARM LIMIT AT ABSOLUTE ENERGY ERROR AVERAGE EneErAvMax
      100
>ALGO_ALARM SUPERIOR LIMIT AT LVDT LvdtMax (1,NCon)
      1e10  1e10
>ALGO_ALARM INFERIOR LIMIT AT LVDT LvdtMin (1,NCon)
      -1e10 -1e10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION1 Press1Max (1,NCon)
      230   230
>ALGO_ALARM INFERIOR LIMIT AT PRESSION1 Press1Min (1,NCon)
      -10   -10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION2 Press2Max (1,NCon)
      230   230
>ALGO_ALARM INFERIOR LIMIT AT PRESSION2 Press2Min (1,NCon)
      -10   -10
>ALGO_ALARM SUPERIOR LIMIT AT SERVOVALVE ServoMax (1,NCon)
      5     5
>ALGO_ALARM INFERIOR LIMIT AT SERVOVALVE ServoMin (1,NCon)
      -5    -5
```

C. Ground acceleration file cent_acc.txt

```
>>>Input ground acceleration history
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line

>TITLE OF THE RECORD:
   El Centro 3.4175 m/s/s, 20 s, NEFOREEE specified
>NUMBER OF RECORD POINTS OF THIS HISTORY  NRecGAcc:
1001
>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS TimeRecIncr s:
      0.02
>ACCELERATION VALUES GAcc (NRecGAcc,1) m/s/s:
0
-0.014002968
-0.10801309
-0.101011606
(…)
```

EXAMPLE M20

0.20g earthquake 2DoF 4con

A. Data input file m20_dat.txt

```
>>>Data of the test
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line
>>>
>>>----------------------General Data-------------------------------
>>>
>TEST NAME: m20
>TITLE DESCRIBING THE TEST:
0.20g earthquake 2 DoF

>NUMBER OF SLAVE CONTROLLERS CONSIDERED AT THIS MASTER NCon>0:
      4

>EXTERNAL BOARD NUMBER OF EVERY SLAVE CONTROLLER CtrNum (1,NCon):
      1     2     3     4

>PISTON SECTION1 (TENSION CHAMBER) AT Con1 Con2 ...  Section1 (1,NCon)
kN/Bar:
      2.75  2.75  2.75  2.75

>PISTON SECTION2 (COMPRESSION CHAMBER) AT Con1 Con2 ...  Section2 (1,NCon)
kN/Bar:
      2.75  2.75  2.75  2.75

>NUMBER OF PATTERN INPUT FILES NPatt>=0:
      0

>NUMBER OF PSD DEGREES OF FREEDOM NDof>=0:
      2

>IF NDof>0, NUMBER OF GROUND ACCELERATION INPUT FILES NGAcc>=0:
```

```
      1

>IF NDof>0, NUMBER OF STRAIN-RATE DEPENDENT DEVICES TO BE COMPENSATED AT
THE RESTORING FORCES NSR>=0:
      0

>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS AT THE PATTERN AND GROUND
ACELERATION INPUT FILES TimeRecIncr s:
      0.01

>NUMBER OF INTERNAL CONTROLLER 2ms SAMPLINGS BETWEEN TWO RECORDS InterRec:
      2500

>>>Time scale lambda = InterRec*0.002/TimeRecIncr = RealTime/PrototypeTime

>PROTOTYPE TIME FOR NEXT TEST STOP TimeStop s:
      12.5

>>>
>>>-------------------------Pattern Data----------------------------
>>>

>>>Formula for computation of every pattern 1<= M <=NPatt:

>>>IntM(Rec)=IntM(Rec-1) + PattISpanM/100 * PattFileM(Rec-1) * TimeRecIncr

>>>PattM(Rec) = PattSpanM/100 * PattFileM(Rec) + IntM(Rec)

>IF NPatt>0, PROPORTIONAL SPAN PERCENTAGE MULTIPILIER PattSpan (1,NPatt)
%:

>IF NPatt>0, INTEGRAL SPAN PERCENTAGE MULTIPLIER PattISpan (1,NPatt) %/s:

>IF NPatt>0, FILE NAME FOR EVERY PATTERN (UP TO 4 CHARACTERS PER NAME)
PattName (NPatt, 4):

>IF NPatt>0, INFLUENCE MATRIX FROM PATTERNS TO TARGETS Patt2Targ
(NCon,NPatt) (mm OR kN)/(mm OR kN):


>>>
>>>-------------------Other Influence Matrices----------------------
>>>

>INFLUENCE MATRIX FROM HEIDENHAIN TO TARGETS Heid2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0

>INFLUENCE MATRIX FROM TEMPOSONICS TO TARGETS Temp2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0   0.0   0.0   0.0
```

```
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0

>INFLUENCE MATRIX FROM LOAD CELLS TO TARGETS LCell2Targ (NCon,NCon) (mm OR
kN)/kN:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
>INFLUENCE MATRIX FROM FORCE2 CHANNEL TO TARGETS Force22Targ (NCon,NCon)
(mm OR kN)/unit:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0

>INFLUENCE MATRIX FROM SPEED CHANNEL TO TARGETS Speed2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
>INFLUENCE MATRIX FROM LVDT CHANNEL TO TARGETS Lvdt2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
>>>
>>>----------------------PsD equation data----------------------------
>>>


>IF NDof>0, THEORETICAL MASS MATRIX Mass(NDof,NDof) kg:
      23900.0     0.0
      0.0   23650.0


>IF NDof>0, THEORETICAL ADDITIONAL STIFFNESS MATRIX StiffAdd(NDof,NDof)
N/m:
      0.0   0.0
      0.0   0.0


>IF NDof>0, THEORETICAL ADDITIONAL DAMPING MATRIX DampingAdd(NDof,NDof)
Ns/m:
      0.0   0.0
      0.0   0.0

>IF NDof>0, INITIAL DISPLACEMENT DisInit(1,NDof) m:
      0.0   0.0

>IF NDof>0, INITIAL VELOCITY VelInit(1,NDof) m/s:
```

```
      0.0   0.0

>>>To avoid restoring-force offset compensation introduce NFSampl=0

>IF NDof>0, NUMBER OF SAMPLINGS TO AVERAGE FOR RESTORING FORCE OFFSET
COMPUTATION NFSampl:
      5000

>IF NDof>0 AND NFSampl>0, PRESCRIBED RESTORING FORCE VALUE FOR OFFSET
COMPUTATION ResInit(1,NDof) N:
      0.0   0.0

>IF NGacc>0, PROPORTIONAL SPAN PERCENTAGE MULTIPLIER GAccSpan(1,NGAcc) %:
      20.0

>>>Accelerogram time increment must be equal to prototype time increment
>IF NGAcc>0, FILE NAME FOR EVERY GROUND ACCELEROGRAM (UP TO 4 CHARACTERS
PER NAME) GAccName(NGAcc,4):
      esec

>IF NDof>0 AND NGAcc>0, INFLUENCE MATRIX FROM GROUND MOTION TO DoF
GAcc2Dof(NDof,NGAcc) (m/s/s)/(m/s/s):
      1.0
      1.0

>IF NDof>0, INFLUENCE MATRIX FROM LOAD CELLS TO RESTORING FORCES
LCell2Res(NDof,NCon) N/kN:
      1000.0      1000.0      0.0   0.0
      0.0   0.0   1000.0      1000.0


>IF NDof>0, INFLUENCE MATRIX FROM DoF DISPLACEMENTS TO TARGETS
Dis2Targ(NCon,NDof) mm/m:
      1000.0      0.0
      1000.0      0.0
      0.0   1000.0
      0.0   1000.0

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO FSR
Force22FSR(NSR,NCon) kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO FSR Speed2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO FSR Lvdt2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO DSR
Force22DSR(NSR,NCon) mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO DSR Speed2DSR(NSR,NCon)
mm/unit:
```

```
>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO DSR Lvdt2DSR(NSR,NCon)
mm/unit:

>>>Formula for strain-rate-compensation additional force at every device:
>>>FSRAdd = SRFacF0*FSR + SRFacD0*DSR + SRFacF1*FSRdot + SRFacD1*DSRdot

>IF NSR>0, FORCE CORRECTION FACTOR SRFacF0(1,NSR) kN/kN:

>IF NSR>0,  DISPLACEMENT CORRECTION FACTOR SRFacD0(1,NSR) kN/mm:

>IF NSR>0,  FORCE-DERIVATIVE CORRECTION FACTOR SRFacF1(1,NSR) kNs/kN:

>IF NSR>0,  DISPLACEMENT-DERIVATIVE CORRECTION FACTOR SRFacD1(1,NSR)
kNs/mm:

>IF NSR>0, INFLUENCE MATRIX FROM FSRAdd TO RESTORING FORCES
FSR2Res(NDof,NSR) N/kN:

>>>
>>>----------------------Algo Alarm Data-----------------------------
>>>

>ALGO_ALARM SUPERIOR LIMIT AT HEIDENHAIN HeidMax (1,NCon)
      100   100   200   200
>ALGO_ALARM INFERIOR LIMIT AT HEIDENHAIN HeidMin (1,NCon)
      -100  -100  -200  -200
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS TempMax (1,NCon)
      100   100   200   200
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS TempMin (1,NCon)
      -100  -100  -200  -200
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS ABS TempAbsMax (1,NCon)
      800   800   800   800
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS ABS TempAbsMin (1,NCon)
      200   200   200   200
>ALGO_ALARM SUPERIOR LIMIT AT LOAD CELL FORCE LCellMax (1,NCon)
      100   100   100   100
>ALGO_ALARM INFERIOR LIMIT AT LOAD CELL FORCE LCellMin (1,NCon)
      -100  -100  -100  -100
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR ErrorMax (1,NCon)
      0.4   0.4   0.4   0.4
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR AVERAGE ErrAvMax (1,NCon)
      0.1   0.1   0.1   0.1
>ALGO_ALARM LIMIT AT ABSOLUTE ENERGY ERROR AVERAGE EneErAvMax
      100
>ALGO_ALARM SUPERIOR LIMIT AT LVDT LvdtMax (1,NCon)
      1e10  1e10  1e10  1e10
>ALGO_ALARM INFERIOR LIMIT AT LVDT LvdtMin (1,NCon)
      -1e10 -1e10 -1e10 -1e10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION1 Press1Max (1,NCon)
      230   230   230   230
>ALGO_ALARM INFERIOR LIMIT AT PRESSION1 Press1Min (1,NCon)
      -10   -10   -10   -10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION2 Press2Max (1,NCon)
      230   230   230   230
```

```
>ALGO_ALARM INFERIOR LIMIT AT PRESSION2 Press2Min (1,NCon)
      -10   -10   -10   -10
>ALGO_ALARM SUPERIOR LIMIT AT SERVOVALVE ServoMax (1,NCon)
      5     5     5     5
>ALGO_ALARM INFERIOR LIMIT AT SERVOVALVE ServoMin (1,NCon)
      -5    -5    -5    -5
```

B. Ground acceleration file esec_acc.txt

```
>>>Input ground acceleration history
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line

>TITLE OF THE RECORD:
    ESECMaSE scaled 1.00g

>NUMBER OF RECORD POINTS OF THIS HISTORY  NRecGAcc:
    1024
>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS TimeRecIncr s:
    0.01
>ACCELERATION VALUES GAcc (NRecGAcc,1) m/s/s:
0
0
0
(…)
```

EXAMPLE M09

PsD for Snap Back m06 4DoF 4con

A. Data input file m09_dat.txt

```
>>>Data of the test
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line
>>>
>>>----------------------General Data-------------------------------
>>>
>TEST NAME: m09
>TITLE DESCRIBING THE TEST:
PsD for Snap Back m06 4 DoF

>NUMBER OF SLAVE CONTROLLERS CONSIDERED AT THIS MASTER NCon>0:
      4

>EXTERNAL BOARD NUMBER OF EVERY SLAVE CONTROLLER CtrNum (1,NCon):
      1     2     3     4

>PISTON SECTION1 (TENSION CHAMBER) AT Con1 Con2 ...  Section1 (1,NCon)
kN/Bar:
      2.75  2.75  2.75  2.75

>PISTON SECTION2 (COMPRESSION CHAMBER) AT Con1 Con2 ...  Section2 (1,NCon)
kN/Bar:
      2.75  2.75  2.75  2.75

>NUMBER OF PATTERN INPUT FILES NPatt>=0:
      0

>NUMBER OF PSD DEGREES OF FREEDOM NDof>=0:
      4

>IF NDof>0, NUMBER OF GROUND ACCELERATION INPUT FILES NGAcc>=0:
```

```
      0

>IF NDof>0, NUMBER OF STRAIN-RATE DEPENDENT DEVICES TO BE COMPENSATED AT
THE RESTORING FORCES NSR>=0:
      0

>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS AT THE PATTERN AND GROUND
ACELERATION INPUT FILES TimeRecIncr s:
      0.002

>NUMBER OF INTERNAL CONTROLLER 2ms SAMPLINGS BETWEEN TWO RECORDS InterRec:
      1000

>>>Time scale lambda = InterRec*0.002/TimeRecIncr = RealTime/PrototypeTime

>PROTOTYPE TIME FOR NEXT TEST STOP TimeStop s:
      2e-3

>>>
>>>-------------------------Pattern Data----------------------------
>>>

>>>Formula for computation of every pattern 1<= M <=NPatt:

>>>IntM(Rec)=IntM(Rec-1) + PattISpanM/100 * PattFileM(Rec-1) * TimeRecIncr

>>>PattM(Rec) = PattSpanM/100 * PattFileM(Rec) + IntM(Rec)

>IF NPatt>0, PROPORTIONAL SPAN PERCENTAGE MULTIPILIER PattSpan (1,NPatt)
%:

>IF NPatt>0, INTEGRAL SPAN PERCENTAGE MULTIPLIER PattISpan (1,NPatt) %/s:

>IF NPatt>0, FILE NAME FOR EVERY PATTERN (UP TO 4 CHARACTERS PER NAME)
PattName (NPatt, 4):

>IF NPatt>0, INFLUENCE MATRIX FROM PATTERNS TO TARGETS Patt2Targ
(NCon,NPatt) (mm OR kN)/(mm OR kN):

>>>
>>>-------------------Other Influence Matrices----------------------
>>>

>INFLUENCE MATRIX FROM HEIDENHAIN TO TARGETS Heid2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0

>INFLUENCE MATRIX FROM TEMPOSONICS TO TARGETS Temp2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
```

```
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0

>INFLUENCE MATRIX FROM LOAD CELLS TO TARGETS LCell2Targ (NCon,NCon) (mm OR
kN)/kN:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
>INFLUENCE MATRIX FROM FORCE2 CHANNEL TO TARGETS Force22Targ (NCon,NCon)
(mm OR kN)/unit:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0

>INFLUENCE MATRIX FROM SPEED CHANNEL TO TARGETS Speed2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
>INFLUENCE MATRIX FROM LVDT CHANNEL TO TARGETS Lvdt2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
>>>
>>>----------------------PsD equation data----------------------------
>>>

>IF NDof>0, THEORETICAL MASS MATRIX Mass(NDof,NDof) kg:
 7.559831e+003 1.887955e+003 5.797518e+002 -3.878690e+001
 1.887955e+003 6.108595e+003 -4.367190e+001 5.906942e+002
 5.797518e+002 -4.367190e+001 6.207467e+003 1.439219e+003
 -3.878690e+001 5.906942e+002 1.439219e+003 5.378971e+003


>IF NDof>0, THEORETICAL ADDITIONAL STIFFNESS MATRIX StiffAdd(NDof,NDof)
N/m:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0

>IF NDof>0, THEORETICAL ADDITIONAL DAMPING MATRIX DampingAdd(NDof,NDof)
Ns/m:
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
      0.0   0.0   0.0   0.0
>IF NDof>0, INITIAL DISPLACEMENT DisInit(1,NDof) m:
      0.3119e-3 0.3180e-3 0.9566e-3 0.9240e-3
```

```
>IF NDof>0, INITIAL VELOCITY VelInit(1,NDof) m/s:
      0.0   0.0   0.0   0.0

>>>To avoid restoring-force offset compensation introduce NFSampl=0

>IF NDof>0, NUMBER OF SAMPLINGS TO AVERAGE FOR RESTORING FORCE OFFSET
COMPUTATION NFSampl:
      5000

>IF NDof>0 AND NFSampl>0, PRESCRIBED RESTORING FORCE VALUE FOR OFFSET
COMPUTATION ResInit(1,NDof) N:
      0.0   0.0   0.0   0.0

>IF NGacc>0, PROPORTIONAL SPAN PERCENTAGE MULTIPLIER GAccSpan(1,NGAcc) %:

>>>Accelerogram time increment must be equal to prototype time increment
>IF NGAcc>0, FILE NAME FOR EVERY GROUND ACCELEROGRAM (UP TO 4 CHARACTERS
PER NAME) GAccName(NGAcc,4):


>IF NDof>0 AND NGAcc>0, INFLUENCE MATRIX FROM GROUND MOTION TO DoF
GAcc2Dof(NDof,NGAcc) (m/s/s)/(m/s/s):

>IF NDof>0, INFLUENCE MATRIX FROM LOAD CELLS TO RESTORING FORCES
LCell2Res(NDof,NCon) N/kN:
      1000.0      0.0   0.0   0.0
      0.0   1000.0      0.0   0.0
      0.0   0.0   1000.0      0.0
      0.0   0.0   0.0   1000.0

>IF NDof>0, INFLUENCE MATRIX FROM DoF DISPLACEMENTS TO TARGETS
Dis2Targ(NCon,NDof) mm/m:
      1000.0      0.0   0.0   0.0
      0.0   1000.0      0.0   0.0
      0.0   0.0   1000.0      0.0
      0.0   0.0   0.0   1000.0

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO FSR
Force22FSR(NSR,NCon) kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO FSR Speed2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO FSR Lvdt2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO DSR
Force22DSR(NSR,NCon) mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO DSR Speed2DSR(NSR,NCon)
mm/unit:
```

```
>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO DSR Lvdt2DSR(NSR,NCon)
mm/unit:

>>>Formula for strain-rate-compensation additional force at every device:
>>>FSRAdd = SRFacF0*FSR + SRFacD0*DSR + SRFacF1*FSRdot + SRFacD1*DSRdot

>IF NSR>0, FORCE CORRECTION FACTOR SRFacF0(1,NSR) kN/kN:

>IF NSR>0,  DISPLACEMENT CORRECTION FACTOR SRFacD0(1,NSR) kN/mm:

>IF NSR>0,  FORCE-DERIVATIVE CORRECTION FACTOR SRFacF1(1,NSR) kNs/kN:

>IF NSR>0,  DISPLACEMENT-DERIVATIVE CORRECTION FACTOR SRFacD1(1,NSR)
kNs/mm:

>IF NSR>0, INFLUENCE MATRIX FROM FSRAdd TO RESTORING FORCES
FSR2Res(NDof,NSR) N/kN:

>>>
>>>----------------------Algo Alarm Data-----------------------------
>>>

>ALGO_ALARM SUPERIOR LIMIT AT HEIDENHAIN HeidMax (1,NCon)
      0.5   0.5   1.0   1.0
>ALGO_ALARM INFERIOR LIMIT AT HEIDENHAIN HeidMin (1,NCon)
      -0.5  -0.5  -1.0  -1.0
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS TempMax (1,NCon)
      1.0   1.0   1.5   1.5
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS TempMin (1,NCon)
      -1.0  -1.0  -1.5  -1.5
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS ABS TempAbsMax (1,NCon)
       496  493   498   483
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS ABS TempAbsMin (1,NCon)
       490  487   492   477
>ALGO_ALARM SUPERIOR LIMIT AT LOAD CELL FORCE LCellMax (1,NCon)
      20    20    20    20
>ALGO_ALARM INFERIOR LIMIT AT LOAD CELL FORCE LCellMin (1,NCon)
      -20   -20   -20   -20
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR ErrorMax (1,NCon)
      0.3   0.3   0.3   0.3
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR AVERAGE ErrAvMax (1,NCon)
      0.1   0.1   0.1   0.1
>ALGO_ALARM LIMIT AT ABSOLUTE ENERGY ERROR AVERAGE EneErAvMax
      100
>ALGO_ALARM SUPERIOR LIMIT AT LVDT LvdtMax (1,NCon)
      1e10  1e10  1e10  1e10
>ALGO_ALARM INFERIOR LIMIT AT LVDT LvdtMin (1,NCon)
      -1e10 -1e10 -1e10 -1e10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION1 Press1Max (1,NCon)
      1e10  1e10  1e10  1e10
>ALGO_ALARM INFERIOR LIMIT AT PRESSION1 Press1Min (1,NCon)
      -1e10 -1e10 -1e10 -1e10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION2 Press2Max (1,NCon)
      1e10  1e10  1e10  1e10
```

```
>ALGO_ALARM INFERIOR LIMIT AT PRESSION2 Press2Min (1,NCon)
      -1e10 -1e10 -1e10 -1e10
>ALGO_ALARM SUPERIOR LIMIT AT SERVOVALVE ServoMax (1,NCon)
      3     3     3     3
>ALGO_ALARM INFERIOR LIMIT AT SERVOVALVE ServoMin (1,NCon)
      -3    -3    -3    -3
```

B. Matlab post-processing m09dbav

```
clear all;
patroot=[labpath '\ESECMaSE'];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  name of the test and general title
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
experiment='m09';
project='ESECMaSE ELSA';
proj_descr=readtit([patroot '\ESECMaSE_project.txt'],project);
disp([project ': ' proj_descr]);
structure='Clay Brick';
struc_descr=readtit([patroot '\ESECMaSE_structures.txt'],structure);
disp([structure ': ' struc_descr]);
exp_descr=readtit([patroot  '\ESECMaSE_experiments.txt'],experiment);
disp([experiment ': ' exp_descr]);
patRaw=[patroot '\Experiments\' experiment '\Results_Raw\']
patTreated=[patroot '\Experiments\' experiment '\Results_Treated\']

patExp=[patroot '\Experiments\' experiment '\']
patRaw=[patExp 'Results_Raw\']
patTreated=[patExp 'Results_Treated\']
patGraphics=[patExp 'Graphics\']
folders2save={'Config_Master' 'Results_Raw' 'Results_Treated' 'Test_Setup'
...
        'Photo' 'Video'} %These folders will be fully copied in DB
pplist={};sppsave={}; %Lists to fill underneath with postp and signals to
save

%
% DLL parameters:
%
Ndof=4;
Ngacc=0;
NCon=4;
Npatt=0;
NLev=2;

%
% Read the data
%
Nsig=82;   %Check at acquisition header file
pwd1=pwd; cd(patRaw);
```

```
fid=fopen(sprintf('%sAV.D01',experiment),'r')
adata=fread(fid,inf,'float');
fclose(fid); cd(pwd1);
np=length(adata)/Nsig
bdata=reshape(adata,Nsig,np)';

%Remove first point
bdata(1,:)=[];
np=size(bdata,1)

mass=[ 7.559831e+003 1.887955e+003 5.797518e+002 -3.878690e+001
 1.887955e+003 6.108595e+003 -4.367190e+001 5.906942e+002
 5.797518e+002 -4.367190e+001 6.207467e+003 1.439219e+003
 -3.878690e+001 5.906942e+002 1.439219e+003 5.378971e+003];  %kg theoretical
specimen mass
Heid2Dof=eye(4)/1000  %Transforms from Heidenhain to DoFs

%
% Signal list literally copied from acquisition header file:
%

% [CHANNELS]
% 1 : TIME
% 2 : ALGORAV.iRecAv  Counter of step record of the accelerogram
% 3 : ALGORAV.InterAv  Number of internal substeps in a record of the
accelerogram
% 4 : ALGORAV.TimeAv  Time
% 5 : ALGORAV.EneAbsAv  Controller Absorbed Energy
% 6 : ALGORAV.EneErrAv  Controller Error Energy
% 7 : ALGORAV.DisAv01  Dof Displacement
% 8 : ALGORAV.DisAv02  Dof Displacement
% 9 : ALGORAV.DisAv03  Dof Displacement
% 10 : ALGORAV.DisAv04  Dof Displacement
% 11 : ALGORAV.VelAv01  Dof Velocity
% 12 : ALGORAV.VelAv02  Dof Velocity
% 13 : ALGORAV.VelAv03  Dof Velocity
% 14 : ALGORAV.VelAv04  Dof Velocity
% 15 : ALGORAV.AccAv01  Dof Acceleration
% 16 : ALGORAV.AccAv02  Dof Acceleration
% 17 : ALGORAV.AccAv03  Dof Acceleration
% 18 : ALGORAV.AccAv04  Dof Acceleration
% 19 : ALGORAV.ResAv01  Dof Restoring Force
% 20 : ALGORAV.ResAv02  Dof Restoring Force
% 21 : ALGORAV.ResAv03  Dof Restoring Force
% 22 : ALGORAV.ResAv04  Dof Restoring Force
% 23 : ALGORAV.ExFAv01  Dof External Force
% 24 : ALGORAV.ExFAv02  Dof External Force
% 25 : ALGORAV.ExFAv03  Dof External Force
% 26 : ALGORAV.ExFAv04  Dof External Force
% 27 : ALGORAV.LCellAv01  Load Cell Force
% 28 : ALGORAV.LCellAv02  Load Cell Force
% 29 : ALGORAV.LCellAv03  Load Cell Force
% 30 : ALGORAV.LCellAv04  Load Cell Force
% 31 : ALGORAV.HeidAv01  Heidenhain Displacement
% 32 : ALGORAV.HeidAv02  Heidenhain Displacement
% 33 : ALGORAV.HeidAv03  Heidenhain Displacement
```

```
% 34 : ALGORAV.HeidAv04  Heidenhain Displacement
% 35 : ALGORAV.TempAv01  Temposonics Displacement
% 36 : ALGORAV.TempAv02  Temposonics Displacement
% 37 : ALGORAV.TempAv03  Temposonics Displacement
% 38 : ALGORAV.TempAv04  Temposonics Displacement
% 39 : ALGORAV.TempAbsAv01  Temposonics Absolute Displacement
% 40 : ALGORAV.TempAbsAv02  Temposonics Absolute Displacement
% 41 : ALGORAV.TempAbsAv03  Temposonics Absolute Displacement
% 42 : ALGORAV.TempAbsAv04  Temposonics Absolute Displacement
% 43 : ALGORAV.SpeedAv01  Speed Channel
% 44 : ALGORAV.SpeedAv02  Speed Channel
% 45 : ALGORAV.SpeedAv03  Speed Channel
% 46 : ALGORAV.SpeedAv04  Speed Channel
% 47 : ALGORAV.LvdtAv01  Lvdt Channel
% 48 : ALGORAV.LvdtAv02  Lvdt Channel
% 49 : ALGORAV.LvdtAv03  Lvdt Channel
% 50 : ALGORAV.LvdtAv04  Lvdt Channel
% 51 : ALGORAV.DisConTargetAv01  Controller Target
% 52 : ALGORAV.DisConTargetAv02  Controller Target
% 53 : ALGORAV.DisConTargetAv03  Controller Target
% 54 : ALGORAV.DisConTargetAv04  Controller Target
% 55 : ALGORAV.Press1Av01  Pressure at tension chamber
% 56 : ALGORAV.Press1Av02  Pressure at tension chamber
% 57 : ALGORAV.Press1Av03  Pressure at tension chamber
% 58 : ALGORAV.Press1Av04  Pressure at tension chamber
% 59 : ALGORAV.Press2Av01  Pressure at compression chamber
% 60 : ALGORAV.Press2Av02  Pressure at compression chamber
% 61 : ALGORAV.Press2Av03  Pressure at compression chamber
% 62 : ALGORAV.Press2Av04  Pressure at compression chamber
% 63 : ALGORAV.PDForAv01  Pressure derived force
% 64 : ALGORAV.PDForAv02  Pressure derived force
% 65 : ALGORAV.PDForAv03  Pressure derived force
% 66 : ALGORAV.PDForAv04  Pressure derived force
% 67 : ALGORAV.ServoAv01  Servovalve Command
% 68 : ALGORAV.ServoAv02  Servovalve Command
% 69 : ALGORAV.ServoAv03  Servovalve Command
% 70 : ALGORAV.ServoAv04  Servovalve Command
% 71 : ALGORAV.SpoolAv01  Servovalve Spool Displacement
% 72 : ALGORAV.SpoolAv02  Servovalve Spool Displacement
% 73 : ALGORAV.SpoolAv03  Servovalve Spool Displacement
% 74 : ALGORAV.SpoolAv04  Servovalve Spool Displacement
% 75 : ALGORAV.ErrAv01  Controller Average Error
% 76 : ALGORAV.ErrAv02  Controller Average Error
% 77 : ALGORAV.ErrAv03  Controller Average Error
% 78 : ALGORAV.ErrAv04  Controller Average Error
% 79 : ALGORAV.ErrMax01  Controller Maximum Error
% 80 : ALGORAV.ErrMax02  Controller Maximum Error
% 81 : ALGORAV.ErrMax03  Controller Maximum Error
% 82 : ALGORAV.ErrMax04  Controller Maximum Error

%
% Define signals' positions
%
mComputerTime=1; miRec=2; mInter=3; mTime=4; k=mTime;

mPatt=k+[1: Npatt]; k=k+Npatt;
```

```
mEneAbs=k+1;k=k+1;
mEneErr=k+1;k=k+1;

mDis=k+[1: Ndof]; k=k+Ndof;
mVel=k+[1: Ndof]; k=k+Ndof;
mAcc=k+[1: Ndof]; k=k+Ndof;
mRes=k+[1: Ndof]; k=k+Ndof;
mExt=k+[1: Ndof];k=k+Ndof;
mGAcc=k+[1:Ngacc];k=k+Ngacc;

mLCell=k+[1: NCon]; k=k+NCon;
mHeid=k+[1: NCon]; k=k+NCon;
mTemp=k+[1: NCon]; k=k+NCon;
mTempAbs=k+[1: NCon];k=k+NCon;
mSpeed=k+[1: NCon]; k=k+NCon;
mLvdt=k+[1: NCon]; k=k+NCon;
mTarget=k+[1: NCon]; k=k+NCon;
mPress1=k+[1: NCon]; k=k+NCon;
mPress2=k+[1: NCon]; k=k+NCon;
mPDFor=k+[1: NCon]; k=k+NCon;
mSV=k+[1: NCon]; k=k+NCon;
mSpool=k+[1: NCon]; k=k+NCon;
mErr=k+[1: NCon]; k=k+NCon;
mErrMax=k+[1: NCon]; k=k+NCon;

if k~=Nsig
    k
    Nsig
    error('Signal ordering error!')
end

%
% Select, reorder and translate into DataBase structure
%
Adata=bdata(:,[mTime mDis mRes mVel mAcc mExt mGAcc]);
s60=sst(cell(1,size(Adata,2)),'Data',Adata);

Adata=bdata(:,[mTime mHeid mLCell mTarget mTemp mTempAbs mErr mErrMax ...
        mPress1 mPress2 mPDFor mSV mSpool mSpeed mLvdt ...
        mEneAbs mEneErr mInter miRec mComputerTime]);
s80=sst(cell(1,size(Adata,2)),'Data',Adata);

% parameters for the identification of eigen frequencies and dampings:
npwin=500;  %dt=0.002   %Number of points for every identification
npjump=10;     %Jump length between two identifications
npol=2*Ndof+[0 2 4];  %Number of poles of the filter model.
nwin=ceil((np-npwin)/npjump)
inittime3=mean(bdata([1 npwin],mTime))
delt=diff(bdata([11 12],mTime))



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Postprocessing general parameters for measured signals
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s=s60;
s=sst(s,'PostProcessing','60');
s=sst(s,'PostP_Descr','PsD Model Measured');
s=sst(s,'Experiment',experiment);
s=sst(s,'Exp_Descr',exp_descr);
s=sst(s,'Structure',structure);
s=sst(s,'Struc_Descr',struc_descr);
s=sst(s,'Project',project);
s=sst(s,'Proj_Descr',proj_descr);
s=sst(s,'Positions',{{'' '' '' '' ''}});

%%%%%%%      TIME  SIGNAL    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=1;m=1;
s(n)=sst(s(n),'Description',''); s(n)=sst(s(n),'Magnitude','Time');
s(n)=sst(s(n),'Unit','s');
repsig(s(n));
nTime60=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ndof];m=m+length(n);
s(n)=sst(s(n),'Description','Reference');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','m');
for i=1:Ndof
 s(n(i))=sst(s(n(i)),'Positions',{{sprintf('Dof %d',i)}},1);
end
s(n(1:2))=sst(s(n(1:2)),'Positions',{{'Level 1'}},2);
s(n(3:4))=sst(s(n(3:4)),'Positions',{{'Level 2'}},2);
s(n([1 3]))=sst(s(n([1 3])),'Positions',{{'S'}},3);
s(n([2 4]))=sst(s(n([2 4])),'Positions',{{'N'}},3);
repsig(s(n));
nRefDisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ndof];m=m+length(n);
s(n)=sst(s(n),'Description','Restoring'); s(n)=sst(s(n),'Magnitude','Force');
s(n)=sst(s(n),'Unit','N');
s(n)=sst(s(n),'Positions',gst(s(nRefDisplacement),'Positions'));
repsig(s(n));
nRestForce=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ndof];m=m+length(n);
s(n)=sst(s(n),'Description',''); s(n)=sst(s(n),'Magnitude','Velocity');
s(n)=sst(s(n),'Unit','m/s');
s(n)=sst(s(n),'Positions',gst(s(nRefDisplacement),'Positions'));
repsig(s(n));
nVelocity=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ndof];m=m+length(n);
s(n)=sst(s(n),'Description','Relative');
s(n)=sst(s(n),'Magnitude','Acceleration');
s(n)=sst(s(n),'Unit','m/s/s');
```

```
s(n)=sst(s(n),'Positions',gst(s(nRefDisplacement),'Positions'));
repsig(s(n));
nRelAcceleration=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ndof];m=m+length(n);
s(n)=sst(s(n),'Description','Equivalent External');
s(n)=sst(s(n),'Magnitude','Force');
s(n)=sst(s(n),'Unit',gst(s(nRestForce),'Unit'));
s(n)=sst(s(n),'Positions',gst(s(nRefDisplacement),'Positions'));
repsig(s(n));
nExForce=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ngacc];m=m+length(n);
if Ngacc>0
    s(n)=sst(s(n),'Description','Ground');
s(n)=sst(s(n),'Magnitude','Acceleration');
    s(n)=sst(s(n),'Unit','m/s/s');
    repsig(s(n));
end
nGAcceleration=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sppsave={sppsave{:} s([nTime60 nRefDisplacement nRestForce nVelocity ...
       nRelAcceleration nGAcceleration])};
pplist={pplist{:} '60'}; %Lists to fill with postp. and signals to save
s60=s;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Postprocessing general parameters for measured signals
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s=s80;
s=sst(s,'PostProcessing','80');
s=sst(s,'PostP_Descr','Controller Measured');
s=sst(s,'Experiment',experiment);
s=sst(s,'Exp_Descr',exp_descr);
s=sst(s,'Structure',structure);
s=sst(s,'Struc_Descr',struc_descr);
s=sst(s,'Project',project);
s=sst(s,'Proj_Descr',proj_descr);
s=sst(s,'Positions',{{'' '' '' '' ''}});

%%%%%%%      TIME  SIGNAL    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=1;m=1;
s(n)=sst(s(n),'Description',''); s(n)=sst(s(n),'Magnitude','Time');
s(n)=sst(s(n),'Unit','s');
repsig(s(n));
nTime80=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
```

```
s(n)=sst(s(n),'Description','Heidenhain');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','mm');
for i=1:Ndof
 s(n(i))=sst(s(n(i)),'Positions',{{sprintf('Con %d',i)}},1);
end
s(n(1:2))=sst(s(n(1:2)),'Positions',{{'Level 1'}},2);
s(n(3:4))=sst(s(n(3:4)),'Positions',{{'Level 2'}},2);
s(n([1 3]))=sst(s(n([1 3])),'Positions',{{'S'}},3);
s(n([2 4]))=sst(s(n([2 4])),'Positions',{{'N'}},3);
repsig(s(n));
nHeidDisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Load Cell'); s(n)=sst(s(n),'Magnitude','Force');
s(n)=sst(s(n),'Unit','kN');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nLCellForce=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Target');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','mm');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nTarg=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Temposonics');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','mm');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nTempDisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Temposonics Absolute');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','mm');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nTempAbsDisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Average Error');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','mm');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nAvErrorDisplacement=n;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Max Error');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','mm');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nMaxErrorDisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Chamber1');
s(n)=sst(s(n),'Magnitude','Pressure');
s(n)=sst(s(n),'Unit','Bar');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nPressure1=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Chamber2');
s(n)=sst(s(n),'Magnitude','Pressure');
s(n)=sst(s(n),'Unit','Bar');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nPressure2=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Pressure Derived');
s(n)=sst(s(n),'Magnitude','Force');
s(n)=sst(s(n),'Unit','kN');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
pmean=1:8; offs=0; sens=1;
s(n)=sst(s(n),'Data',physic(gst(s(n),'Data'),sens,pmean,offs));
repsig(s(n));
nPDForce=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','SV Command');
s(n)=sst(s(n),'Magnitude','Voltage');
s(n)=sst(s(n),'Unit','V');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nSVCommand=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','SV Spool');
s(n)=sst(s(n),'Magnitude','Voltage');
s(n)=sst(s(n),'Unit','V');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nSVSpool=n;
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Channel'); s(n)=sst(s(n),'Magnitude','Speed');
s(n)=sst(s(n),'Unit','V');
repsig(s(n));
nSpeed=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Channel'); s(n)=sst(s(n),'Magnitude','Lvdt');
s(n)=sst(s(n),'Unit','mm');
pmean=[]; offs=0; sens=-2.66;
s(n)=sst(s(n),'Data',physic(gst(s(n),'Data'),sens,pmean,offs));
repsig(s(n));
nLvdt=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:1];m=m+length(n);
s(n)=sst(s(n),'Description','Absorbed'); s(n)=sst(s(n),'Magnitude','Energy');
s(n)=sst(s(n),'Unit','J');
repsig(s(n));
nAbsEnergy=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:1];m=m+length(n);
s(n)=sst(s(n),'Description','Error'); s(n)=sst(s(n),'Magnitude','Energy');
s(n)=sst(s(n),'Unit','J');
repsig(s(n));
nErrEnergy=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:1];m=m+length(n);
s(n)=sst(s(n),'Description','Internal Steps');
s(n)=sst(s(n),'Magnitude','Number');
s(n)=sst(s(n),'Unit','-');
repsig(s(n));
nInterNumber=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:1];m=m+length(n);
s(n)=sst(s(n),'Description','Record'); s(n)=sst(s(n),'Magnitude','Number');
s(n)=sst(s(n),'Unit','-');
repsig(s(n));
nRecNumber=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:1];m=m+length(n);
s(n)=sst(s(n),'Description','Computer'); s(n)=sst(s(n),'Magnitude','Time');
s(n)=sst(s(n),'Unit','s');
s(n)=sst(s(n),'Data',gm(s(n))/1000);
repsig(s(n));
nComputerTime=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
sppsave={sppsave{:} s([nTime80 nHeidDisplacement nLCellForce ...
        nAvErrorDisplacement nMaxErrorDisplacement ...
        nInterNumber nRecNumber nComputerTime])};


pplist={pplist{:} '80'}; %Lists to fill with postp. and signals to save
s80=s;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Postprocessing general parameters for derived signals
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s=s60(1);
s=sst(s,'PostProcessing','62');
s=sst(s,'PostP_Descr','PsD Model Derived');

%%%%%%%      TIME  SIGNAL    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=1;m=1;
s(n)=sst(s(n),'Description',''); s(n)=sst(s(n),'Magnitude','Time');
s(n)=sst(s(n),'Unit','s');
repsig(s(n));
nTime62=n;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:4];m=m+length(n);
s(n)=s(ones(1,length(n)));
s(n)=sst(s(n),'Description','Total'); s(n)=sst(s(n),'Magnitude','Energy');
s(n)=sst(s(n),'Unit','J');
s(n(1))=sst(s(n(1)),'Positions',{{'Absorbed'}});
s(n(1))=sst(s(n(1)),'Data',sum(work(gm(s60(nRestForce)),gm(s60(nRefDisplaceme
nt))),2));
s(n(2))=sst(s(n(2)),'Positions',{{'Kinetic'}});
s(n(2))=sst(s(n(2)),'Data',sum(gm(s60(nVelocity)).*(gm(s60(nVelocity))*(mass/
2)),2));
s(n(3))=sst(s(n(3)),'Positions',{{'Absorbed+Kinetic'}});
s(n(3))=sst(s(n(3)),'Data',gm(s(n(1)))+gm(s(n(2))));
s(n(4))=sst(s(n(4)),'Positions',{{'Input'}});
s(n(4))=sst(s(n(4)),'Data',sum(work(gm(s60(nExForce)),gm(s60(nRefDisplacement
))),2));
repsig(s(n));
nTotalEnergy=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ndof];m=m+length(n);
s(n)=s(ones(1,length(n)));
s(n)=sst(s(n),'Description','Absolute');
s(n)=sst(s(n),'Magnitude','Acceleration');
s(n)=sst(s(n),'Unit','m/s/s');
s(n)=sst(s(n),'Positions',gst(s60(nRefDisplacement),'Positions'));
s(n)=sst(s(n),'Positions',{{'-rest force/mass'}},2);
s(n)=sst(s(n),'Data',((-inv(mass))*(gm(s60(nRestForce))'))');
repsig(s(n));
nAbsoluteAcceleration=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if Ngacc==1;
    n=m+[1:Ndof];m=m+length(n);
    s(n)=s(ones(1,length(n)));
    s(n)=sst(s(n),'Description','Absolute');
s(n)=sst(s(n),'Magnitude','Acceleration');
    s(n)=sst(s(n),'Unit','m/s/s');
    s(n)=sst(s(n),'Positions',gst(s60(nRefDisplacement),'Positions'));
    s(n)=sst(s(n),'Positions',{{'ddu/dt2+ag'}},2);
%
s(n)=sst(s(n),'Data',gm(s60(nRelAcceleration))+gm(s60(nGAcceleration(ones(1,N
Dof)))));

s(n)=sst(s(n),'Data',dericen(gm(s60(nVelocity)),gm(s60(nTime60)))+gm(s60(nGAc
celeration(ones(1,Ndof)))));
    repsig(s(n));
    nAbsoluteAcceleration2=n;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 sppsave={sppsave{:} s([nTime62 nTotalEnergy nAbsoluteAcceleration])};
pplist={pplist{:} '62'}; %Lists to fill with postproc. and signals to save
s62=s;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Postprocessing general parameters for derived signals
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s=s60(1);
s=sst(s,'PostProcessing','63');
s=sst(s,'PostP_Descr','PsD Model Identified');

%%%%%%%      TIME  SIGNAL    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=1;m=1;
s(n)=sst(s(n),'Description',''); s(n)=sst(s(n),'Magnitude','Time');
s(n)=sst(s(n),'Unit','s');
s(n)=sst(s(n),'Data',inittime3+delt*npjump*[0:(nwin-1)]');
repsig(s(n));
nTime63=n;


%Direct identification of k and c
%
disp('Direct identification of k and c and eigen frequencies and dampings');

T=gm(s60(nTime60)); % time in sec
D=gm(s60(nRefDisplacement)); % displ in m
V=dericen(D,T);
HD=gm(s80(nHeidDisplacement));
DM=(Heid2Dof*(gm(s80(nHeidDisplacement))'))';
VM=dericen(DM,T);
R=gm(s60(nRestForce)); % force in N
DV1t=[D V ones(size(D,1),1)];
DMVM1t=[DM VM ones(size(D,1),1)];
```

```
Rt=R;
freq=zeros(nwin,Ndof);freqM=freq;
zeta=zeros(nwin,Ndof);zetaM=zeta;
for iwin=1:nwin;
    pp=[1:npwin]+npjump*(iwin-1);
    %Solving:  [D V 1] [k'; c'; Of'] = R
    kco=(DV1t(pp,:)\Rt(pp,:))';
    k=kco(:,1:Ndof);
    c=kco(:,Ndof+[1:Ndof]);
    foff=kco(:,2*Ndof+1);
    [polesHz]=cmodes(k,c,mass);
    freq(iwin,:)=polesHz(1:Ndof,1)';
    zeta(iwin,:)=100*polesHz(1:Ndof,2)';
    kco=(DMVM1t(pp,:)\Rt(pp,:))';
    km=kco(:,1:Ndof);
    cm=kco(:,Ndof+[1:Ndof]);
    [polesHz]=cmodes(km,cm,mass);
    freqM(iwin,:)=polesHz(1:Ndof,1)';
    zetaM(iwin,:)=100*polesHz(1:Ndof,2)';
end;
%

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:2*Ndof];m=m+length(n);
s(n)=s(ones(1,length(n)));   %Creation of new derived signals
s(n)=sst(s(n),'Description','Spatial-Model');
s(n)=sst(s(n),'Magnitude','Frequency');
s(n)=sst(s(n),'Unit','Hz');
for ic=1:Ndof;
    s(n([0 Ndof]+ic))=sst(s(n([0 Ndof]+ic)),'Positions', ...
        {{sprintf('Mode %d',ic)}},1);
end;
s(n(1:Ndof))=sst(s(n(1:Ndof)),'Positions',{{'RefD'}},2);
s(n(Ndof+[1:Ndof]))=sst(s(n(Ndof+[1:Ndof])),'Positions',{{'MeaD'}},2);
s(n)=sst(s(n),'Positions',{{sprintf('w=%d',npwin)}},3);
s(n)=sst(s(n),'Data',[freq freqM]);
repsig(s(n));
nSMFrequency=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:2*Ndof];m=m+length(n);
s(n)=s(ones(1,length(n)));   %Creation of new derived signals
s(n)=s(nSMFrequency);
s(n)=sst(s(n),'Description','Spatial-Model');
s(n)=sst(s(n),'Magnitude','Damping Ratio');
s(n)=sst(s(n),'Unit','%');
s(n)=sst(s(n),'Data',[zeta zetaM]);
repsig(s(n));
nSMDamping=n;

%
% Filter model
%
disp('Filter Model: Identification of eigen frequencies and dampings');
```

```
npolm=max(npol)/2;
nid=length(npol);
cpo=zeros(nwin,nid*npolm);
GAcc=cell2mat(gst(s60(nGAcceleration),'Data'));
RDisp=cell2mat(gst(s60(nRefDisplacement),'Data'));
for iwin=1:nwin;
    pp=[1 npwin]+npjump*(iwin-1);
    d=RDisp(pp(1):pp(2),:);
    if Ngacc>0
        ag=GAcc([pp(1):pp(2)],:);
        if all(ag==0);
            ag=zeros(npwin,0);
        end;
    else
        ag=zeros(npwin,0);
    end
    for i1=1:nid;
        [polesHz,modes,cpoles]=timepole(d,ag,delt,npol(i1));
        cpo(iwin,(i1-1)*npolm+[1:npol(i1)/2])=cpoles(1:npol(i1)/2)';
    end;
end;
ii0=npolm*[0:nid-1];
ii=range2([1:Ndof],ii0);
ii=ii(:)';

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:(nid*npolm)];m=m+length(n);
s(n)=s(ones(1,length(n)));   %Creation of new derived signals
s(n)=sst(s(n),'Description','Filter-Model');
s(n)=sst(s(n),'Magnitude','Frequency');
s(n)=sst(s(n),'Unit','Hz');
for i=1:nid;
    s(n((i-1)*npolm+[1:npolm]))=sst(s(n((i-1)*npolm+[1:npolm])), ...
        'Positions',{{sprintf('%dpol',npol(i))}},2);
    s(n((i-1)*npolm+[1:npolm]))=sst(s(n((i-1)*npolm+[1:npolm])), ...
        'Positions',{{sprintf('w=%d',npwin)}},3);
    sprintf('%dpol w=%d',npol(i),npwin)
    for ic=1:Ndof;
        s(n((i-1)*npolm+ic))=sst(s(n((i-1)*npolm+ic)),'Positions', ...
            {{sprintf('Mode %d',ic)}},1);
    end;
end;
s(n)=sst(s(n),'Data',abs(cpo)/2/pi);
repsig(s(n));
nFMFrequency=n;

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:(nid*npolm)];m=m+length(n);
s(n)=s(ones(1,length(n)));   %Creation of new derived signals
s(n)=s(nFMFrequency);
s(n)=sst(s(n),'Description','Filter-Model');
s(n)=sst(s(n),'Magnitude','Damping Ratio');
s(n)=sst(s(n),'Unit','%');
s(n)=sst(s(n),'Data',-100*cos(angle(cpo)));
repsig(s(n));
nFMDamping=n;
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1];m=m+length(n);
s(n)=s(ones(1,length(n)));   %Creation of new derived signals
s(n)=sst(s(n),'Description','Av. Amplitude Top');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','m');
% s(n)=sst(s(n),'Positions',{{sprintf('Level %d',Ndof)}},1);
s(n)=sst(s(n),'Positions',{{'RefD'}},2);
s(n)=sst(s(n),'Positions',{{sprintf('w=%d',npwin)}},3);
D=gm(s60(nRefDisplacement(Ndof)));
Dcomplx=fhilbt(D);
ampl_D=smooth(abs(Dcomplx),wwin(npwin,'r'),1);
s(n)=sst(s(n),'Data',ampl_D);
s(n)=rformst(s(n),npjump,npwin/2,npwin/2+npjump*nwin);
repsig(s(n));
nAVADisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 sppsave={sppsave{:} s([nTime63 nSMFrequency nSMDamping ...
             nFMFrequency(1:Ndof) nFMDamping(1:Ndof) nAVADisplacement])};
pplist={pplist{:} '63'}; %Lists to fill with postproc. and signals to save
s63=s;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Postprocessing general parameters for derived signals
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s=s80(1);
s=sst(s,'PostProcessing','82');
s=sst(s,'PostP_Descr','Controller Derived');

%%%%%%%      TIME  SIGNAL    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=1;m=1;
s(n)=sst(s(n),'Description',''); s(n)=sst(s(n),'Magnitude','Time');
s(n)=sst(s(n),'Unit','s');
repsig(s(n));
nTime82=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=s(ones(1,length(n)));
s(n)=sst(s(n),'Description','Heidenhain Drift');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','mm');
s(n)=sst(s(n),'Positions',gst(s80(nHeidDisplacement),'Positions'));
s(n)=sst(s(n),'Positions',{{''}},1);
s(n([1 3]))=sst(s(n([1 3])),'Data',isdrift(gm(s80(nHeidDisplacement([1
3])))));
s(n([2 4]))=sst(s(n([2 4])),'Data',isdrift(gm(s80(nHeidDisplacement([2
4])))));
repsig(s(n));
nHeidDriftDispl=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
n=m+[1:NCon];m=m+length(n);
s(n)=s(ones(1,length(n)));
s(n)=sst(s(n),'Description','LCell Shear');
s(n)=sst(s(n),'Magnitude','Force');
s(n)=sst(s(n),'Unit','kN');
s(n)=sst(s(n),'Positions',gst(s(nHeidDriftDispl),'Positions'));
s(n([1 3]))=sst(s(n([1 3])),'Data',shearlo(gm(s80(nLCellForce([1 3])))));
s(n([2 4]))=sst(s(n([2 4])),'Data',shearlo(gm(s80(nLCellForce([2 4])))));
repsig(s(n));
nLCellShearForce=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NLev];m=m+length(n);
s(n)=s(ones(1,length(n)));
s(n)=sst(s(n),'Description','Controller I-S Absorbed');
s(n)=sst(s(n),'Magnitude','Energy');
s(n)=sst(s(n),'Unit','J');
s(n)=sst(s(n),'Positions',gst(s(nHeidDriftDispl),'Positions'));
s(n)=sst(s(n),'Data',work(gm(s(nLCellShearForce)),gm(s(nHeidDriftDispl))));
repsig(s(n));
nConAbsorbedEnergy=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 sppsave={sppsave{:} s([nTime82 nHeidDriftDispl nLCellShearForce
nConAbsorbedEnergy])};
pplist={pplist{:} '82'}; %Lists to fill with postproc. and signals to save
s82=s;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% graphics

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all;
Time60=selst(s60,'Magnitude','Time')
GAcc=selst(selst(s60,'Description','Ground'),'Magnitude','Acceleration')
RDisp=selst(selst(s60,'Description','Reference'),'Magnitude','Displacement')
Vel=selst(s60,'Magnitude','Velocity')
RForce=selst(selst(s60,'Description','Restoring'),'Magnitude','Force')

Time62=selst(s62,'Magnitude','Time')
ECForce=selst(selst(s82,'Description','Ext. Cell'),'Magnitude','Force')
RForce62=selst(selst(s62,'Description','Restoring'),'Magnitude','Force')
MDisp=selst(selst(s62,'Description','Mean'),'Magnitude','Displacement')
ISDrift=selst(selst(s62,'Description','I-S
Drift'),'Magnitude','Displacement')
TEner=selst(selst(s62,'Description','Total'),'Magnitude','Energy')
AAcc=selst(selst(s62,'Description','Absolute'),'Magnitude','Acceleration')

Time63=selst(s63,'Magnitude','Time')
SMFreq=selst(selst(s63,'Description','Spatial-
Model'),'Magnitude','Frequency')
SMDamp=selst(selst(s63,'Description','Spatial-Model'),'Magnitude','Damping
Ratio')
SMFreq2=SMFreq((Ndof+1):(2*Ndof))
```

```
SMFreq=SMFreq(1:Ndof)
SMDamp2=SMDamp((Ndof+1):(2*Ndof))
SMDamp=SMDamp(1:Ndof)
FMFreq=selst(selst(s63,'Description','Filter-Model'),'Magnitude','Frequency')
FMDamp=selst(selst(s63,'Description','Filter-Model'),'Magnitude','Damping
Ratio')
AVADisp=selst(selst(s63,'Description','Av. Amplitude
Top'),'Magnitude','Displacement')


Time80=selst(s80,'Magnitude','Time');
Time80=Time80(1)
HDisp=selst(selst(s80,'Description','Heidenhain'),'Magnitude','Displacement')
LCForce=selst(selst(s80,'Description','Load Cell'),'Magnitude','Force')
TDisp=selst(selst(s80,'Description','Temposonics'),'Magnitude','Displacement'
)
TAbsDisp=selst(selst(s80,'Description','Temposonics
Absolute'),'Magnitude','Displacement')
AEner=selst(selst(s80,'Description','Absorbed'),'Magnitude','Energy')
EEner=selst(selst(s80,'Description','Error'),'Magnitude','Energy')
EDisp=selst(selst(s80,'Description','Average
Error'),'Magnitude','Displacement')
EMDisp=selst(selst(s80,'Description','Max Error'),'Magnitude','Displacement')
Press1=selst(selst(s80,'Description','Chamber1'),'Magnitude','Pressure')
Press2=selst(selst(s80,'Description','Chamber2'),'Magnitude','Pressure')
PDForce=selst(selst(s80,'Description','Pressure
Derived'),'Magnitude','Force')
SV=selst(selst(s80,'Description','SV Command'),'Magnitude','Voltage')
InterN=selst(selst(s80,'Description','Internal Steps'),'Magnitude','Number')

Time82=selst(s82,'Magnitude','Time')
HeidDrift=selst(selst(s82,'Description','Heidenhain
Drift'),'Magnitude','Displacement')
LCellShearForce=selst(selst(s82,'Description','LCell
Shear'),'Magnitude','Force')
nConAbsorbedEnergy=selst(selst(s82,'Description','Controller I-S
Absorbed'),'Magnitude','Energy')

%
% Graphs to save [
%
n=0; descrlist={};
%

if Ngacc>0
    n=n+1;descrlist{n}='Ground Acceleration Histories'; g=[];
    g=[];  g.pl{1}.ysig=GAcc;g.pl{1}.xsig=Time60;
    m=[]; m.gr={g} ; mgra(m,'l',n);  set(n,'Name', descrlist{n});
end

n=n+1; descrlist{n}='Algorithm Displacement Histories'; g=[];
g.pl{1}.ysig={RDisp{1:Ndof}};g.pl{1}.xsig=Time60;
g.pl{2}.ysig={RForce{1:Ndof}};g.pl{2}.xsig=Time60;
m=[]; m.gr={g} ; mgra(m,'p',n); set(n,'Name', descrlist{n});
```

```
n=n+1; descrlist{n}='Frequency and damping (measured and performed)'; m=[];
for i=1:2;
    m.gr{1,i}.pl(1)=sst(sst(cell(1,1),'ysig',{{SMFreq{i}
SMFreq2{i}}}),'xsig',{Time63});
    m.gr{2,i}.pl(1)=sst(sst(cell(1,1),'ysig',{{SMDamp{i}
SMDamp2{i}}}),'xsig',{Time63});
end
mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Frequency and damping (measured and performed)'; m=[];
for i=3:4;
    m.gr{1,i-2}.pl(1)=sst(sst(cell(1,1),'ysig',{{SMFreq{i}
SMFreq2{i}}}),'xsig',{Time63});
    m.gr{2,i-2}.pl(1)=sst(sst(cell(1,1),'ysig',{{SMDamp{i}
SMDamp2{i}}}),'xsig',{Time63});
end
mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Displacement amplitud / frequency or damping';
m=[]; m.gr=sst(cell(2,1),'pl',{ ...
        sst(sst(cell(1,1),'ysig',{SMFreq{1}}),'xsig',{{AVADisp{1}}}) ...
        sst(sst(cell(1,1),'ysig',{SMDamp{1}}),'xsig',{{AVADisp{1}}})...
    }); mgra(m,'p',n); set(n,'Name', descrlist{n});

 % n=n+1; descrlist{n}='Controller Force & Displacement Histories'; g=[];
% g.pl{1}.ysig={HDisp{1:NCon}};g.pl{1}.xsig=Time80;
% g.pl{2}.ysig={LCForce{1:NCon}};g.pl{2}.xsig=Time80;
% m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

  n=n+1; descrlist{n}='Shear Force/Drift  Cycle'; g=[];
      g.pl{1}.ysig={LCellShearForce{1:2}};g.pl{1}.xsig={HeidDrift{1:2}};
      g.pl{2}.ysig={LCellShearForce{3:4}};g.pl{2}.xsig={HeidDrift{3:4}};
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

mgra(m,'p',n); set(n,'Name', descrlist{n});


descrl=descrlist; %This list of graphs will be saved in database
%
%   ] End of graphs to save.
%
%
% Graphs not to save [
%

n=100; descrlist={};

 n=n+1; descrlist{n}='Heidenhain vs Temposonics Displacement Histories';
g=[];
 g.pl{1}.ysig={HDisp{1:2} TDisp{1:2}};g.pl{1}.xsig=Time80;
  g.pl{2}.ysig={HDisp{3:4} TDisp{3:4}};g.pl{2}.xsig=Time80;
 m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});


n=n+1; descrlist{n}='Frequency and damping (check by two models)'; m=[];
```

```
for i1=1:2;
    i=i1;
    m.gr{1,i1}.pl(1)=sst(sst(cell(1,1),'ysig',{{FMFreq{ii0+i}
SMFreq{i}}}),'xsig',{Time63});
    m.gr{2,i1}.pl(1)=sst(sst(cell(1,1),'ysig',{{FMDamp{ii0+i}
SMDamp{i}}}),'xsig',{Time63});
end
mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Frequency and damping (check by two models)'; m=[];
for i1=3:Ndof;
    i=i1;
    m.gr{1,i1-2}.pl(1)=sst(sst(cell(1,1),'ysig',{{FMFreq{ii0+i}
SMFreq{i}}}),'xsig',{Time63});
    m.gr{2,i1-2}.pl(1)=sst(sst(cell(1,1),'ysig',{{FMDamp{ii0+i}
SMDamp{i}}}),'xsig',{Time63});
end
mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Temposonics Displacement Histories'; g=[];
g.pl{1}.ysig={TDisp{1:NCon}};g.pl{1}.xsig=Time80;
g.pl{2}.ysig={TAbsDisp{1:NCon}};g.pl{2}.xsig=Time80;
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Controller Error Histories'; g=[];
g.pl{1}.ysig={EDisp{1:NCon}};g.pl{1}.xsig=Time80;
g.pl{2}.ysig={EMDisp{1:NCon}};g.pl{2}.xsig=Time80;
g.pl{3}.ysig={InterN{:}};g.pl{3}.xsig=Time80;
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Controller Error Cycles'; g=[];
g.pl{1}.ysig={EDisp{1} EDisp{2} EDisp{3} EDisp{4}};g.pl{1}.xsig={Vel{1}
Vel{1} Vel{2} Vel{2}};
g.pl{2}.ysig={EMDisp{1:NCon}};g.pl{2}.xsig={LCForce{1:NCon}};
g.pl{3}.ysig={InterN{:}};g.pl{3}.xsig=Time80;
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Controller Energy Histories'; g=[];
g.pl{1}.ysig={AEner{:} EEner{:}};g.pl{1}.xsig=Time80;
g.pl{2}.ysig={EEner{:}};g.pl{2}.xsig=Time80;
g.pl{3}.ysig={InterN{:}};g.pl{3}.xsig=Time80;
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});


n=n+1; descrlist{n}='Controller Servo Valve Histories'; g=[];
for i=1:NCon
  g.pl{i}.ysig={SV{i}};g.pl{i}.xsig=Time80;
end
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Algorithm Energy and Acc. Histories'; g=[];
g.pl{1}.ysig={TEner{:}};g.pl{1}.xsig=Time82;
g.pl{2}.ysig={AAcc{:}};g.pl{2}.xsig=Time82;
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  write the treated files in LAB folder and in DB
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
put_treat(pplist,sppsave,patExp,patTreated,patGraphics,descrl, ...
     folders2save,project,structure,experiment);
```

EXAMPLE e23

bench test 2acc 2con 2DoF

A. Data input file e23_dat.txt

```
>>>Data of the test
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line
>>>
>>>----------------------General Data-------------------------------
>>>
>TEST NAME: e23
>TITLE DESCRIBING THE TEST:
bench test 2 accelerogram 2 slave controllers 2DoF

>NUMBER OF SLAVE CONTROLLERS CONSIDERED AT THIS MASTER NCon>0:
      2

>EXTERNAL BOARD NUMBER OF EVERY SLAVE CONTROLLER CtrNum (1,NCon):
      1     2

>PISTON SECTION1 (TENSION CHAMBER) AT Con1 Con2 ...  Section1 (1,NCon)
kN/Bar:
      2.75  2.75

>PISTON SECTION2 (COMPRESSION CHAMBER) AT Con1 Con2 ...  Section2 (1,NCon)
kN/Bar:
      2.75  2.75

>NUMBER OF PATTERN INPUT FILES NPatt>=0:
      0

>NUMBER OF PSD DEGREES OF FREEDOM NDof>=0:
      2

>IF NDof>0, NUMBER OF GROUND ACCELERATION INPUT FILES NGAcc>=0:
```

```
      2

>IF NDof>0, NUMBER OF STRAIN-RATE DEPENDENT DEVICES TO BE COMPENSATED AT
THE RESTORING FORCES NSR>=0:
      0

>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS AT THE PATTERN AND GROUND
ACELERATION INPUT FILES TimeRecIncr s:
      0.02

>NUMBER OF INTERNAL CONTROLLER 2ms SAMPLINGS BETWEEN TWO RECORDS InterRec:
      10

>>>Time scale lambda = InterRec*0.002/TimeRecIncr = RealTime/PrototypeTime

>PROTOTYPE TIME FOR NEXT TEST STOP TimeStop s:
      30.0

>>>
>>>-------------------------Pattern Data----------------------------
>>>

>>>Formula for computation of every pattern 1<= M <=NPatt:

>>>IntM(Rec)=IntM(Rec-1) + PattISpanM/100 * PattFileM(Rec-1) * TimeRecIncr

>>>PattM(Rec) = PattSpanM/100 * PattFileM(Rec) + IntM(Rec)

>IF NPatt>0, PROPORTIONAL SPAN PERCENTAGE MULTIPILIER PattSpan (1,NPatt)
%:

>IF NPatt>0, INTEGRAL SPAN PERCENTAGE MULTIPLIER PattISpan (1,NPatt) %/s:

>IF NPatt>0, FILE NAME FOR EVERY PATTERN (UP TO 4 CHARACTERS PER NAME)
PattName (NPatt, 4):

>IF NPatt>0, INFLUENCE MATRIX FROM PATTERNS TO TARGETS Patt2Targ
(NCon,NPatt) (mm OR kN)/(mm OR kN):

>>>
>>>-------------------Other Influence Matrices----------------------
>>>

>INFLUENCE MATRIX FROM HEIDENHAIN TO TARGETS Heid2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM TEMPOSONICS TO TARGETS Temp2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0   0.0
      0.0   0.0
```

```
>INFLUENCE MATRIX FROM LOAD CELLS TO TARGETS LCell2Targ (NCon,NCon) (mm OR
kN)/kN:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM FORCE2 CHANNEL TO TARGETS Force22Targ (NCon,NCon)
(mm OR kN)/unit:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM SPEED CHANNEL TO TARGETS Speed2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0   0.0
      0.0   0.0
>INFLUENCE MATRIX FROM LVDT CHANNEL TO TARGETS Lvdt2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0   0.0
      0.0   0.0

>>>
>>>----------------------PsD equation data---------------------------
>>>

>IF NDof>0, THEORETICAL MASS MATRIX Mass(NDof,NDof) kg:
      8300.0      0.0
      0.0   5.0e6


>IF NDof>0, THEORETICAL ADDITIONAL STIFFNESS MATRIX StiffAdd(NDof,NDof)
N/m:
      2.05e6      0.0
      0.0   2.0e8


>IF NDof>0, THEORETICAL ADDITIONAL DAMPING MATRIX DampingAdd(NDof,NDof)
Ns/m:
      0.0   0.0
      0.0   0.0

>IF NDof>0, INITIAL DISPLACEMENT DisInit(1,NDof) m:
      0.0   0.0

>IF NDof>0, INITIAL VELOCITY VelInit(1,NDof) m/s:
      0.0   0.0

>>>To avoid restoring-force offset compensation introduce NFSampl=0

>IF NDof>0, NUMBER OF SAMPLINGS TO AVERAGE FOR RESTORING FORCE OFFSET
COMPUTATION NFSampl:
      5000

>IF NDof>0 AND NFSampl>0, PRESCRIBED RESTORING FORCE VALUE FOR OFFSET
COMPUTATION ResInit(1,NDof) N:
      0.0   0.0
```

```
>IF NGacc>0, PROPORTIONAL SPAN PERCENTAGE MULTIPLIER GAccSpan(1,NGAcc) %:
      0.06  0.06

>>>Accelerogram time increment must be equal to prototype time increment
>IF NGAcc>0, FILE NAME FOR EVERY GROUND ACCELEROGRAM (UP TO 4 CHARACTERS
PER NAME) GAccName(NGAcc,4):
      octu
      quab

>IF NDof>0 AND NGAcc>0, INFLUENCE MATRIX FROM GROUND MOTION TO DoF
GAcc2Dof(NDof,NGAcc) (m/s/s)/(m/s/s):
      1.0   0.0
      0.0   1.0

>IF NDof>0, INFLUENCE MATRIX FROM LOAD CELLS TO RESTORING FORCES
LCell2Res(NDof,NCon) N/kN:
      1000.0      1000.0
      -1886.5     1886.5


>IF NDof>0, INFLUENCE MATRIX FROM DoF DISPLACEMENTS TO TARGETS
Dis2Targ(NCon,NDof) mm/m:
      1000.0      -1377.5
      1000.0      1377.5

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO FSR
Force22FSR(NSR,NCon) kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO FSR Speed2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO FSR Lvdt2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO DSR
Force22DSR(NSR,NCon) mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO DSR Speed2DSR(NSR,NCon)
mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO DSR Lvdt2DSR(NSR,NCon)
mm/unit:

>>>Formula for strain-rate-compensation additional force at every device:
>>>FSRAdd = SRFacF0*FSR + SRFacD0*DSR + SRFacF1*FSRdot + SRFacD1*DSRdot

>IF NSR>0, FORCE CORRECTION FACTOR SRFacF0(1,NSR) kN/kN:

>IF NSR>0,  DISPLACEMENT CORRECTION FACTOR SRFacD0(1,NSR) kN/mm:

>IF NSR>0,  FORCE-DERIVATIVE CORRECTION FACTOR SRFacF1(1,NSR) kNs/kN:
```

```
>IF NSR>0,  DISPLACEMENT-DERIVATIVE CORRECTION FACTOR SRFacD1(1,NSR)
kNs/mm:

>IF NSR>0, INFLUENCE MATRIX FROM FSRAdd TO RESTORING FORCES
FSR2Res(NDof,NSR) N/kN:

>>>
>>>----------------------Algo Alarm Data-----------------------------
>>>

>ALGO_ALARM SUPERIOR LIMIT AT HEIDENHAIN HeidMax (1,NCon)
      2     2
>ALGO_ALARM INFERIOR LIMIT AT HEIDENHAIN HeidMin (1,NCon)
      -2    -2
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS TempMax (1,NCon)
      30    30
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS TempMin (1,NCon)
      -30   -30
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS ABS TempAbsMax (1,NCon)
      1e10  1e10
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS ABS TempAbsMin (1,NCon)
      -1e10 -1e10
>ALGO_ALARM SUPERIOR LIMIT AT LOAD CELL FORCE LCellMax (1,NCon)
      40    40
>ALGO_ALARM INFERIOR LIMIT AT LOAD CELL FORCE LCellMin (1,NCon)
      -40   -40
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR ErrorMax (1,NCon)
      30    30
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR AVERAGE ErrAvMax (1,NCon)
      1e10  1e10
>ALGO_ALARM LIMIT AT ABSOLUTE ENERGY ERROR AVERAGE EneErAvMax
      1e10
>ALGO_ALARM SUPERIOR LIMIT AT LVDT LvdtMax (1,NCon)
      1e10  1e10
>ALGO_ALARM INFERIOR LIMIT AT LVDT LvdtMin (1,NCon)
      -1e10 -1e10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION1 Press1Max (1,NCon)
      230   230
>ALGO_ALARM INFERIOR LIMIT AT PRESSION1 Press1Min (1,NCon)
      -10   -10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION2 Press2Max (1,NCon)
      230   230
>ALGO_ALARM INFERIOR LIMIT AT PRESSION2 Press2Min (1,NCon)
      -10   -10
>ALGO_ALARM SUPERIOR LIMIT AT SERVOVALVE ServoMax (1,NCon)
      10    10
>ALGO_ALARM INFERIOR LIMIT AT SERVOVALVE ServoMin (1,NCon)
      -10   -10
```

B. Ground acceleration file quab_acc.txt

```
>>>Input ground acceleration history
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line
>TITLE OF THE RECORD:
   Precon QuakeB 1.00g
>NUMBER OF RECORD POINTS OF THIS HISTORY  NRecGAcc:
1802
>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS TimeRecIncr s:
      0.02
>ACCELERATION VALUES GAcc (NRecGAcc,1) m/s/s:
 -2.6094600e-002
 -2.5898400e-002
 -2.5996500e-002
(…)
```

C. Matlab post-processing e23dbav.m

```
clear all;
patroot=[labpath '\BenchTest'];
%patroot=['\\Smdc01\labfolders\LAB\BenchTest'];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  name of the test and general title
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
experiment='e23';
project='xxxx';
proj_descr=readtit([patroot '\BenchTest_project.txt'],project);
disp([project ': ' proj_descr]);
structure='nefo';
struc_descr=readtit([patroot '\BenchTest_structures.txt'],structure);
disp([structure ': ' struc_descr]);
exp_descr=readtit([patroot  '\BenchTest_experiments.txt'],experiment);
disp([experiment ': ' exp_descr]);
patRaw=[patroot '\Experiments\' experiment '\Results_Raw\']
patTreated=[patroot '\Experiments\' experiment '\Results_Treated\']

patExp=[patroot '\Experiments\' experiment '\']
patRaw=[patExp 'Results_Raw\']
patTreated=[patExp 'Results_Treated\']
patGraphics=[patExp 'Graphics\']
folders2save={'Config_Master' 'Results_Raw' 'Results_Treated' 'Test_Setup'
...
        'Photo' 'Video'} %These folders will be fully copied in DB
pplist={};sppsave={}; %Lists to fill underneath with postp and signals to
save

%
% DLL parameters:
```

```
%
Ndof=2;
Ngacc=2;
NCon=2;
Npatt=0;

%
% Read the data
%
Nsig=46;   %Check at acquisition header file
pwd1=pwd; cd(patRaw);
fid=fopen(sprintf('%sAV.D07',experiment),'r')
adata=fread(fid,inf,'float');
fclose(fid); cd(pwd1);
np=length(adata)/Nsig
bdata=reshape(adata,Nsig,np)';

mass=[8300 0.0; 0.0 5.0e6];  %kg design mass
GAcc2Dof=[1.0 0.0;0.0 1.0 ];
AccIni=0;
VelIni=0;
DisIni=0;
DampAdd=[0];
StifAdd=[   2.05e6  0.0;0.0     2.0e8];
LCell2Res=[ 1000.0  1000.0; -1886.5 1886.5];
GAccSpan=[1.0 10.0];
Dis2Targ=[1000.0    -1377.5;    1000.0  1377.5]

%
% Signal list literally copied from acquisition header file:
%

% [CHANNELS]
% 1 : TIME
% 2 : ALGORAV.iRecAv  Counter of step record of the accelerogram
% 3 : ALGORAV.InterAv  Number of internal substeps in a record of the
accelerogram
% 4 : ALGORAV.TimeAv  Time
% 5 : ALGORAV.EneAbsAv  Controller Absorbed Energy
% 6 : ALGORAV.EneErrAv  Controller Error Energy
% 7 : ALGORAV.DisAv01  Dof Displacement
% 8 : ALGORAV.DisAv02  Dof Displacement
% 9 : ALGORAV.VelAv01  Dof Velocity
% 10 : ALGORAV.VelAv02  Dof Velocity
% 11 : ALGORAV.AccAv01  Dof Acceleration
% 12 : ALGORAV.AccAv02  Dof Acceleration
% 13 : ALGORAV.ResAv01  Dof Restoring Force
% 14 : ALGORAV.ResAv02  Dof Restoring Force
% 15 : ALGORAV.ExFAv01  Dof External Force
% 16 : ALGORAV.ExFAv02  Dof External Force
% 17 : ALGORAV.GAccAv01  Scaled Ground Acceleration
% 18 : ALGORAV.GAccAv02  Scaled Ground Acceleration
% 19 : ALGORAV.LCellAv01  Load Cell Force
% 20 : ALGORAV.LCellAv02  Load Cell Force
% 21 : ALGORAV.HeidAv01  Heidenhain Displacement
% 22 : ALGORAV.HeidAv02  Heidenhain Displacement
```

```
% 23 : ALGORAV.TempAv01  Temposonics Displacement
% 24 : ALGORAV.TempAv02  Temposonics Displacement
% 25 : ALGORAV.TempAbsAv01  Temposonics Absolute Displacement
% 26 : ALGORAV.TempAbsAv02  Temposonics Absolute Displacement
% 27 : ALGORAV.SpeedAv01  Speed Channel
% 28 : ALGORAV.SpeedAv02  Speed Channel
% 29 : ALGORAV.LvdtAv01  Lvdt Channel
% 30 : ALGORAV.LvdtAv02  Lvdt Channel
% 31 : ALGORAV.DisConTargetAv01  Controller Target
% 32 : ALGORAV.DisConTargetAv02  Controller Target
% 33 : ALGORAV.Press1Av01  Pressure at tension chamber
% 34 : ALGORAV.Press1Av02  Pressure at tension chamber
% 35 : ALGORAV.Press2Av01  Pressure at compression chamber
% 36 : ALGORAV.Press2Av02  Pressure at compression chamber
% 37 : ALGORAV.PDForAv01  Pressure derived force
% 38 : ALGORAV.PDForAv02  Pressure derived force
% 39 : ALGORAV.ServoAv01  Servovalve Command
% 40 : ALGORAV.ServoAv02  Servovalve Command
% 41 : ALGORAV.SpoolAv01  Servovalve Spool Displacement
% 42 : ALGORAV.SpoolAv02  Servovalve Spool Displacement
% 43 : ALGORAV.ErrAv01  Controller Average Error
% 44 : ALGORAV.ErrAv02  Controller Average Error
% 45 : ALGORAV.ErrMax01  Controller Maximum Error
% 46 : ALGORAV.ErrMax02  Controller Maximum Error

%
% Define signals' positions
%
nComputerTime=1; niRec=2; nInter=3; nTime=4; k=nTime;

nPatt=k+[1: Npatt]; k=k+Npatt;
nEneAbs=k+1;k=k+1;
nEneErr=k+1;k=k+1;

nDis=k+[1: Ndof]; k=k+Ndof;
nVel=k+[1: Ndof]; k=k+Ndof;
nAcc=k+[1: Ndof]; k=k+Ndof;
nRes=k+[1: Ndof]; k=k+Ndof;
nExt=k+[1: Ndof];k=k+Ndof;
nGAcc=k+[1:Ngacc];k=k+Ngacc;

nLCell=k+[1: NCon]; k=k+NCon;
nHeid=k+[1: NCon]; k=k+NCon;
nTemp=k+[1: NCon]; k=k+NCon;
nTempAbs=k+[1: NCon];k=k+NCon;
nSpeed=k+[1: NCon]; k=k+NCon;
nLvdt=k+[1: NCon]; k=k+NCon;
nTarget=k+[1: NCon]; k=k+NCon;
nPress1=k+[1: NCon]; k=k+NCon;
nPress2=k+[1: NCon]; k=k+NCon;
nPDFor=k+[1: NCon]; k=k+NCon;
nSV=k+[1: NCon]; k=k+NCon;
nSpool=k+[1: NCon]; k=k+NCon;
nErr=k+[1: NCon]; k=k+NCon;
nErrMax=k+[1: NCon]; k=k+NCon;
```

```
%
% Select, reorder and translate into DataBase structure
%
Adata=bdata(:,[nTime nDis nRes nVel nAcc nExt nGAcc]);
s60=sst(cell(1,size(Adata,2)),'Data',Adata);

Adata=bdata(:,[nTime nSpeed nLvdt nTarget]);
s70=sst(cell(1,size(Adata,2)),'Data',Adata);

Adata=bdata(:,[nTime nHeid nLCell nTemp nTempAbs nErr nErrMax ...
        nPress1 nPress2 nPDFor nSV nSpool ...
        nEneAbs nEneErr nInter niRec nComputerTime]);
s80=sst(cell(1,size(Adata,2)),'Data',Adata);


% parameters for the identification of eigen frequencies and dampings:
npwin=100;      %Number of points for every identification
npjump=10;      %Jump length between two identifications
npol=2*Ndof+[0 2 4];  %Number of poles of the filter model.
nwin=ceil((np-npwin)/npjump)
inittime3=mean(bdata([1 npwin],nTime))
delt=diff(bdata([1 2],nTime))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Postprocessing general parameters for measured signals
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s=s60;
s=sst(s,'PostProcessing','60');
s=sst(s,'PostP_Descr','PsD Model Measured');
s=sst(s,'Experiment',experiment);
s=sst(s,'Exp_Descr',exp_descr);
s=sst(s,'Structure',structure);
s=sst(s,'Struc_Descr',struc_descr);
s=sst(s,'Project',project);
s=sst(s,'Proj_Descr',proj_descr);
s=sst(s,'Positions',{{'' '' '' '' ''}});

%%%%%%%      TIME  SIGNAL    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=1;m=1;
s(n)=sst(s(n),'Description',''); s(n)=sst(s(n),'Magnitude','Time');
s(n)=sst(s(n),'Unit','s');
repsig(s(n));
nTime60=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ndof];m=m+length(n);
s(n)=sst(s(n),'Description','Reference');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','m');
for i=1:NCon
  s(n(i))=sst(s(n(i)),'Positions',{{sprintf('Dof %d',i)}},1);
end
repsig(s(n));
```

```
nRefDisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ndof];m=m+length(n);
s(n)=sst(s(n),'Description','Restoring'); s(n)=sst(s(n),'Magnitude','Force');
s(n)=sst(s(n),'Unit','N');
for i=1:NCon
  s(n(i))=sst(s(n(i)),'Positions',{{sprintf('Dof %d',i)}},1);
end
repsig(s(n));
nRestForce=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ndof];m=m+length(n);
s(n)=sst(s(n),'Description',''); s(n)=sst(s(n),'Magnitude','Velocity');
s(n)=sst(s(n),'Unit','m/s');
s(n)=sst(s(n),'Positions',gst(s(nRefDisplacement),'Positions'));
repsig(s(n));
nVelocity=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ndof];m=m+length(n);
s(n)=sst(s(n),'Description','Relative');
s(n)=sst(s(n),'Magnitude','Acceleration');
s(n)=sst(s(n),'Unit','m/s/s');
s(n)=sst(s(n),'Positions',gst(s(nRefDisplacement),'Positions'));
repsig(s(n));
nRelAcceleration=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ndof];m=m+length(n);
s(n)=sst(s(n),'Description','Equivalent External');
s(n)=sst(s(n),'Magnitude','Force');
s(n)=sst(s(n),'Unit',gst(s(nRestForce),'Unit'));
s(n)=sst(s(n),'Positions',gst(s(nRefDisplacement),'Positions'));
repsig(s(n));
nExForce=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:Ngacc];m=m+length(n);
s(n)=sst(s(n),'Description','Ground');
s(n)=sst(s(n),'Magnitude','Acceleration');
s(n)=sst(s(n),'Unit','m/s/s');
for i=1:NCon
  s(n(i))=sst(s(n(i)),'Positions',{{sprintf('Accelerogram%d',i)}},1);
end
repsig(s(n));
nGAcceleration=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sppsave={sppsave{:} s([nTime60 nRefDisplacement nRestForce nVelocity ...
        nRelAcceleration nGAcceleration])};
pplist={pplist{:} '60'}; %Lists to fill with postp. and signals to save
s60=s;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Postprocessing general parameters for measured signals
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s=s70;
s=sst(s,'PostProcessing','70');
s=sst(s,'PostP_Descr','Channels');
s=sst(s,'Experiment',experiment);
s=sst(s,'Exp_Descr',exp_descr);
s=sst(s,'Structure',structure);
s=sst(s,'Struc_Descr',struc_descr);
s=sst(s,'Project',project);
s=sst(s,'Proj_Descr',proj_descr);
s=sst(s,'Positions',{{'' '' '' '' ''}});

%%%%%%%      TIME  SIGNAL    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=1;m=1;
s(n)=sst(s(n),'Description',''); s(n)=sst(s(n),'Magnitude','Time');
s(n)=sst(s(n),'Unit','s');
repsig(s(n));
nTime70=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Channel'); s(n)=sst(s(n),'Magnitude','Speed');
s(n)=sst(s(n),'Unit','V');
repsig(s(n));
nSpeed=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Channel'); s(n)=sst(s(n),'Magnitude','Lvdt');
s(n)=sst(s(n),'Unit','mm');
pmean=[]; offs=0; sens=-2.66;
s(n)=sst(s(n),'Data',physic(gst(s(n),'Data'),sens,pmean,offs));
repsig(s(n));
nLvdt=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Target');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','m');
repsig(s(n));
nTarg=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sppsave={sppsave{:} s([nTime70 nSpeed nLvdt])};
pplist={pplist{:} '70'}; %Lists to fill with postp. and signals to save
s70=s;
%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%
%  Postprocessing general parameters for measured signals
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s=s80;
s=sst(s,'PostProcessing','80');
s=sst(s,'PostP_Descr','Controller Measured');
s=sst(s,'Experiment',experiment);
s=sst(s,'Exp_Descr',exp_descr);
s=sst(s,'Structure',structure);
s=sst(s,'Struc_Descr',struc_descr);
s=sst(s,'Project',project);
s=sst(s,'Proj_Descr',proj_descr);
s=sst(s,'Positions',{{'' '' '' '' ''}});

%%%%%%%      TIME  SIGNAL    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=1;m=1;
s(n)=sst(s(n),'Description',''); s(n)=sst(s(n),'Magnitude','Time');
s(n)=sst(s(n),'Unit','s');
repsig(s(n));
nTime80=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Heidenhain');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','mm');
for i=1:NCon
  s(n(i))=sst(s(n(i)),'Positions',{{sprintf('Con %d',i)}},1);
end
repsig(s(n));
nHeidDisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Load Cell'); s(n)=sst(s(n),'Magnitude','Force');
s(n)=sst(s(n),'Unit','kN');
for i=1:NCon
  s(n(i))=sst(s(n(i)),'Positions',{{sprintf('Con %d',i)}},1);
end
repsig(s(n));
nLCellForce=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Temposonics');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','mm');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nTempDisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Temposonics Absolute');
s(n)=sst(s(n),'Magnitude','Displacement');
```

```
s(n)=sst(s(n),'Unit','mm');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nTempAbsDisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Average Error');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','mm');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nAvErrorDisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Max Error');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','mm');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nMaxErrorDisplacement=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Chamber1');
s(n)=sst(s(n),'Magnitude','Pressure');
s(n)=sst(s(n),'Unit','Bar');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nPressure1=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Chamber2');
s(n)=sst(s(n),'Magnitude','Pressure');
s(n)=sst(s(n),'Unit','Bar');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nPressure2=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','Pressure Derived');
s(n)=sst(s(n),'Magnitude','Force');
s(n)=sst(s(n),'Unit','kN');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
pmean=1:8; offs=0; sens=1;
s(n)=sst(s(n),'Data',physic(gst(s(n),'Data'),sens,pmean,offs));
repsig(s(n));
nPDForce=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','SV Command');
s(n)=sst(s(n),'Magnitude','Voltage');
```

```
s(n)=sst(s(n),'Unit','V');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nSVCommand=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:NCon];m=m+length(n);
s(n)=sst(s(n),'Description','SV Spool');
s(n)=sst(s(n),'Magnitude','Voltage');
s(n)=sst(s(n),'Unit','V');
s(n)=sst(s(n),'Positions',gst(s(nHeidDisplacement),'Positions'));
repsig(s(n));
nSVSpool=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:1];m=m+length(n);
s(n)=sst(s(n),'Description','Absorbed'); s(n)=sst(s(n),'Magnitude','Energy');
s(n)=sst(s(n),'Unit','J');
repsig(s(n));
nAbsEnergy=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:1];m=m+length(n);
s(n)=sst(s(n),'Description','Error'); s(n)=sst(s(n),'Magnitude','Energy');
s(n)=sst(s(n),'Unit','J');
repsig(s(n));
nErrEnergy=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:1];m=m+length(n);
s(n)=sst(s(n),'Description','Internal Steps');
s(n)=sst(s(n),'Magnitude','Number');
s(n)=sst(s(n),'Unit','-');
repsig(s(n));
nInterNumber=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:1];m=m+length(n);
s(n)=sst(s(n),'Description','Record'); s(n)=sst(s(n),'Magnitude','Number');
s(n)=sst(s(n),'Unit','-');
repsig(s(n));
nRecNumber=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:1];m=m+length(n);
s(n)=sst(s(n),'Description','Computer'); s(n)=sst(s(n),'Magnitude','Time');
s(n)=sst(s(n),'Unit','s');
s(n)=sst(s(n),'Data',gm(s(n))/1000);
repsig(s(n));
nComputerTime=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sppsave={sppsave{:} s([nTime80 nHeidDisplacement nLCellForce
nTempDisplacement ...
        nTempAbsDisplacement nAvErrorDisplacement nMaxErrorDisplacement ...
```

```
        nInterNumber nRecNumber nComputerTime])};
pplist={pplist{:} '80'}; %Lists to fill with postp. and signals to save
s80=s;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Postprocessing general parameters for derived signals
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s=s60(1);
s=sst(s,'PostProcessing','63');
s=sst(s,'PostP_Descr','PsD Model Identified');

%%%%%%%      TIME  SIGNAL    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=1;m=1;
s(n)=sst(s(n),'Description',''); s(n)=sst(s(n),'Magnitude','Time');
s(n)=sst(s(n),'Unit','s');
s(n)=sst(s(n),'Data',inittime3+delt*npjump*[0:(nwin-1)]');
repsig(s(n));
nTime63=n;


%Direct identification of k and c
%
disp('Direct identification of k and c and eigen frequencies and dampings');

T=gm(s60(nTime60)); % time in sec
D=gm(s60(nRefDisplacement)); % displ in m
V=dericen(D,T);
HD=gm(s80(nHeidDisplacement));
DM=Dis2Targ*(gm(s80(nHeidDisplacement))');
DM=DM'/1000;
VM=dericen(DM,T);
R=gm(s60(nRestForce)); % force in N
pw=1;
DV1t=smooth([D V ones(size(D,1),1)],wwin(pw,'h'),1);
DMVM1t=smooth([DM VM ones(size(D,1),1)],wwin(pw,'h'),1);
Rt=smooth(R,wwin(pw,'h'),1);
freq=zeros(nwin,Ndof);freqM=freq;
zeta=zeros(nwin,Ndof);zetaM=zeta;
for iwin=1:nwin;
    pp=[1:npwin]+npjump*(iwin-1);
    %Solving:  [D V 1] [k'; c'; Of'] = R
    kco=(DV1t(pp,:)\Rt(pp,:))';
    k=kco(:,1:Ndof);
    c=kco(:,Ndof+[1:Ndof]);
    foff=kco(:,2*Ndof+1);
    [polesHz]=cmodes(k,c,mass);
    freq(iwin,:)=polesHz(1:Ndof,1)';
    zeta(iwin,:)=100*polesHz(1:Ndof,2)';
    kco=(DMVM1t(pp,:)\Rt(pp,:))';
    km=kco(:,1:Ndof);
    cm=kco(:,Ndof+[1:Ndof]);
    [polesHz]=cmodes(km,cm,mass);
    freqM(iwin,:)=polesHz(1:Ndof,1)';
    zetaM(iwin,:)=100*polesHz(1:Ndof,2)';
```

```
end;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:2*Ndof];m=m+length(n);
s(n)=s(ones(1,length(n)));   %Creation of new derived signals
s(n)=sst(s(n),'Description','Spatial-Model');
s(n)=sst(s(n),'Magnitude','Frequency');
s(n)=sst(s(n),'Unit','Hz');
for ic=1:Ndof;
    s(n([0 Ndof]+ic))=sst(s(n([0 Ndof]+ic)),'Positions', ...
        {{sprintf('Mode %d',ic)}},1);
end;
s(n(1:Ndof))=sst(s(n(1:Ndof)), ...
    'Positions',{{sprintf('RefD w=%d',npwin)}},2);
s(n(Ndof+[1:Ndof]))=sst(s(n(Ndof+[1:Ndof])), ...
    'Positions',{{sprintf('MeaD w=%d',npwin)}},2);
s(n)=sst(s(n),'Data',[freq freqM]);
repsig(s(n));
nSMFrequency=n;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:2*Ndof];m=m+length(n);
s(n)=s(ones(1,length(n)));   %Creation of new derived signals
s(n)=s(nSMFrequency);
s(n)=sst(s(n),'Description','Spatial-Model');
s(n)=sst(s(n),'Magnitude','Damping Ratio');
s(n)=sst(s(n),'Unit','%');
s(n)=sst(s(n),'Data',[zeta zetaM]);
repsig(s(n));
nSMDamping=n;

% %
% Filter model
%
disp('Filter Model: Identification of eigen frequencies and dampings');

npolm=max(npol)/2;
nid=length(npol);
cpo=zeros(nwin,nid*npolm);
GAcc=cell2mat(gst(s60(nGAcceleration),'Data'));
RDisp=cell2mat(gst(s60(nRefDisplacement),'Data'));
for iwin=1:nwin;
    pp=[1 npwin]+npjump*(iwin-1);
    ag=GAcc([pp(1):pp(2)],:); d=RDisp(pp(1):pp(2),:);
    if all(ag==0);
        ag=zeros(npwin,0);
    end;
    for i1=1:nid;
        [polesHz,modes,cpoles]=timepole(d,ag,delt,npol(i1));
        cpo(iwin,(i1-1)*npolm+[1:npol(i1)/2])=cpoles(1:npol(i1)/2)';
    end;
end;
ii0=npolm*[0:nid-1];
ii=range2([1:Ndof],ii0);
ii=ii(:)';
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:(nid*npolm)];m=m+length(n);
s(n)=s(ones(1,length(n)));   %Creation of new derived signals
s(n)=sst(s(n),'Description','Filter-Model');
s(n)=sst(s(n),'Magnitude','Frequency');
s(n)=sst(s(n),'Unit','Hz');
for i=1:nid;
    s(n((i-1)*npolm+[1:npolm]))=sst(s(n((i-1)*npolm+[1:npolm])), ...
        'Positions',{{sprintf('%dpol w=%d',npol(i),npwin)}},2);
    sprintf('%dpol w=%d',npol(i),npwin)
    for ic=1:Ndof;
        s(n((i-1)*npolm+ic))=sst(s(n((i-1)*npolm+ic)),'Positions', ...
            {{sprintf('Mode %d',ic)}},1);
    end;
end;
s(n)=sst(s(n),'Data',abs(cpo)/2/pi);
repsig(s(n));
nFMFrequency=n;

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1:(nid*npolm)];m=m+length(n);
s(n)=s(ones(1,length(n)));   %Creation of new derived signals
s(n)=s(nFMFrequency);
s(n)=sst(s(n),'Description','Filter-Model');
s(n)=sst(s(n),'Magnitude','Damping Ratio');
s(n)=sst(s(n),'Unit','%');
s(n)=sst(s(n),'Data',-100*cos(angle(cpo)));
repsig(s(n));
nFMDamping=n;

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=m+[1];m=m+length(n);
s(n)=s(ones(1,length(n)));   %Creation of new derived signals
s(n)=sst(s(n),'Description','Av. Amplitude');
s(n)=sst(s(n),'Magnitude','Displacement');
s(n)=sst(s(n),'Unit','m');
% s(n)=sst(s(n),'Positions',{{sprintf('Level %d',Ndof)}},1);
s(n)=sst(s(n),'Positions',{{sprintf('RefD w=%d',npwin)}},2);
D=gm(s60(nRefDisplacement(Ndof)));
Dcomplx=fhilbt(D);
ampl_D=smooth(abs(Dcomplx),wwin(npwin,'r'),1);
s(n)=sst(s(n),'Data',ampl_D);
s(n)=rformst(s(n),npjump,npwin/2,npwin/2+npjump*nwin);
repsig(s(n));
nAVADisplacement=n;

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sppsave={sppsave{:} s([nTime63 nSMFrequency nSMDamping nAVADisplacement])};
pplist={pplist{:} '63'}; %Lists to fill with postproc. and signals to save
s63=s;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% graphics

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
close all;
Time60=selst(s60,'Magnitude','Time')
GAcc=selst(selst(s60,'Description','Ground'),'Magnitude','Acceleration')
RDisp=selst(selst(s60,'Description','Reference'),'Magnitude','Displacement')
Vel=selst(s60,'Magnitude','Velocity')
RForce=selst(selst(s60,'Description','Restoring'),'Magnitude','Force')

Time70=selst(s70,'Magnitude','Time')
Speed=selst(selst(s70,'Description','Channel'),'Magnitude','Speed')
Lvdt=selst(selst(s70,'Description','Channel'),'Magnitude','Lvdt')
Targ=selst(selst(s60,'Description','Target'),'Magnitude','Displacement')

Time80=selst(s80,'Magnitude','Time');
Time80=Time80(1)
HDisp=selst(selst(s80,'Description','Heidenhain'),'Magnitude','Displacement')
LCForce=selst(selst(s80,'Description','Load Cell'),'Magnitude','Force')
AEner=selst(selst(s80,'Description','Absorbed'),'Magnitude','Energy')
EEner=selst(selst(s80,'Description','Error'),'Magnitude','Energy')
EDisp=selst(selst(s80,'Description','Average
Error'),'Magnitude','Displacement')
EMDisp=selst(selst(s80,'Description','Max Error'),'Magnitude','Displacement')
Press1=selst(selst(s80,'Description','Chamber1'),'Magnitude','Pressure')
Press2=selst(selst(s80,'Description','Chamber2'),'Magnitude','Pressure')
PDForce=selst(selst(s80,'Description','Pressure
Derived'),'Magnitude','Force')
SV=selst(selst(s80,'Description','SV Command'),'Magnitude','Voltage')
InterN=selst(selst(s80,'Description','Internal Steps'),'Magnitude','Number')

Time63=selst(s63,'Magnitude','Time')
SMFreq=selst(selst(s63,'Description','Spatial-
Model'),'Magnitude','Frequency')
SMDamp=selst(selst(s63,'Description','Spatial-Model'),'Magnitude','Damping
Ratio')
SMFreq2=SMFreq((Ndof+1):(2*Ndof))
SMFreq=SMFreq(1:Ndof)
SMDamp2=SMDamp((Ndof+1):(2*Ndof))
SMDamp=SMDamp(1:Ndof)
FMFreq=selst(selst(s63,'Description','Filter-Model'),'Magnitude','Frequency')
FMDamp=selst(selst(s63,'Description','Filter-Model'),'Magnitude','Damping
Ratio')
AVADisp=selst(selst(s63,'Description','Av.
Amplitude'),'Magnitude','Displacement')

%
% Graphs to save [
%
n=0; descrlist={};
%

n=n+1; descrlist{n}='Algorithm Displacement Histories'; g=[];
g.pl{1}.ysig={RDisp{1:Ndof}};g.pl{1}.xsig=Time60;
g.pl{2}.ysig={RForce{1:Ndof}};g.pl{2}.xsig=Time60;
m=[]; m.gr={g} ; mgra(m,'p',n); set(n,'Name', descrlist{n});

%
n=n+1; descrlist{n}='Algorithm Force/Displacement Cycle'; g=[];
```

```
g.pl{1}.ysig={RForce{1:Ndof}};g.pl{1}.xsig={RDisp{1:Ndof}};
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Controller Force & Displacement Histories'; g=[];
g.pl{1}.ysig={HDisp{1:NCon}};g.pl{1}.xsig=Time80;
g.pl{2}.ysig={LCForce{1:NCon}};g.pl{2}.xsig=Time80;
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Controller Force/Displacement Cycle'; g=[];
g.pl{1}.ysig={LCForce{1:NCon}};g.pl{1}.xsig={HDisp{1:NCon}};
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Frequency and damping (measured and performed)'; g=[];
g.pl{1}.ysig={SMFreq{:} SMFreq2{:}};g.pl{1}.xsig=Time63;
g.pl{2}.ysig={SMDamp{:} SMDamp2{:}};g.pl{2}.xsig=Time63;
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Displacement amplitud / frequency or damping';
m=[]; m.gr=sst(cell(2,1),'pl',{ ...
        sst(sst(cell(1,1),'ysig',{SMFreq{1}}),'xsig',{{AVADisp{1}}}) ...
        sst(sst(cell(1,1),'ysig',{SMDamp{1}}),'xsig',{{AVADisp{1}}})...
    }); mgra(m,'p',n); set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Frequency and damping (check by two models)'; m=[];
for i1=1:Ndof;
    i=i1;
    m.gr{1,i1}.pl(1)=sst(sst(cell(1,1),'ysig',{{FMFreq{ii0+i}
SMFreq{i}}}),'xsig',{Time63});
    m.gr{2,i1}.pl(1)=sst(sst(cell(1,1),'ysig',{{FMDamp{ii0+i}
SMDamp{i}}}),'xsig',{Time63});
end
mgra(m,'p',n); set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Frequency and damping (check by two models)'; m=[];
for i1=1:Ndof;
    i=i1;
    m.gr{1,i1}.pl(1)=sst(sst(cell(1,1),'ysig',{{FMFreq{ii0+i}
}}),'xsig',{Time63});
end
mgra(m,'p',n); set(n,'Name', descrlist{n});

descrl=descrlist; %This list of graphs will be saved in database
%
%   ] End of graphs to save.
%

%
% Graphs not to save [
%

n=100; descrlist={};

n=n+1;descrlist{n}='Ground Acceleration Histories'; g=[];
g=[];  g.pl{1}.ysig=GAcc;g.pl{1}.xsig=Time60;
m=[]; m.gr={g} ; mgra(m,'l',n);  set(n,'Name', descrlist{n});
```

```
n=n+1; descrlist{n}='Controller Error Histories'; g=[];
g.pl{1}.ysig={EDisp{1:NCon}};g.pl{1}.xsig=Time80;
g.pl{2}.ysig={EMDisp{1:NCon}};g.pl{2}.xsig=Time80;
g.pl{3}.ysig={InterN{:}};g.pl{3}.xsig=Time80;
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Controller Error Cycles'; g=[];
g.pl{1}.ysig={EDisp{1:NCon}};g.pl{1}.xsig=Vel;
g.pl{2}.ysig={EMDisp{1:NCon}};g.pl{2}.xsig={LCForce{1:NCon}};
g.pl{3}.ysig={InterN{:}};g.pl{3}.xsig=Time80;
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Controller Energy Histories'; g=[];
g.pl{1}.ysig={AEner{:} EEner{:}};g.pl{1}.xsig=Time80;
g.pl{2}.ysig={EEner{:}};g.pl{2}.xsig=Time80;
g.pl{3}.ysig={InterN{:}};g.pl{3}.xsig=Time80;
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});

n=n+1; descrlist{n}='Controller Servo Valve Histories'; g=[];
for i=1:NCon
  g.pl{i}.ysig={SV{i}};g.pl{i}.xsig=Time80;
end
m=[]; m.gr={g} ; mgra(m,'p',n);  set(n,'Name', descrlist{n});
```

EXAMPLE e22

cyclic test 2con 2patterns

A. Data input file e22_dat.txt

```
>>>Data of the test
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line
>>>
>>>----------------------General Data-------------------------------
>>>
>TEST NAME: e22
>TITLE DESCRIBING THE TEST:
cyclic test 2 slave controllers

>NUMBER OF SLAVE CONTROLLERS CONSIDERED AT THIS MASTER NCon>0:
      2

>EXTERNAL BOARD NUMBER OF EVERY SLAVE CONTROLLER CtrNum (1,NCon):
      1     2

>PISTON SECTION1 (TENSION CHAMBER) AT Con1 Con2 ...  Section1 (1,NCon)
kN/Bar:
      2.75  2.75

>PISTON SECTION2 (COMPRESSION CHAMBER) AT Con1 Con2 ...  Section2 (1,NCon)
kN/Bar:
      2.75  2.75

>NUMBER OF PATTERN INPUT FILES NPatt>=0:
      2

>NUMBER OF PSD DEGREES OF FREEDOM NDof>=0:
      0

>IF NDof>0, NUMBER OF GROUND ACCELERATION INPUT FILES NGAcc>=0:
```

```
>IF NDof>0, NUMBER OF STRAIN-RATE DEPENDENT DEVICES TO BE COMPENSATED AT
THE RESTORING FORCES NSR>=0:

>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS AT THE PATTERN AND GROUND
ACELERATION INPUT FILES TimeRecIncr s:
      0.02

>NUMBER OF INTERNAL CONTROLLER 2ms SAMPLINGS BETWEEN TWO RECORDS InterRec:
      5000

>>>Time scale lambda = InterRec*0.002/TimeRecIncr = RealTime/PrototypeTime

>PROTOTYPE TIME FOR NEXT TEST STOP TimeStop s:
      30.0

>>>
>>>-------------------------Pattern Data----------------------------
>>>

>>>Formula for computation of every pattern 1<= M <=NPatt:

>>>IntM(Rec)=IntM(Rec-1) + PattISpanM/100 * PattFileM(Rec-1) * TimeRecIncr

>>>PattM(Rec) = PattSpanM/100 * PattFileM(Rec) + IntM(Rec)

>IF NPatt>0, PROPORTIONAL SPAN PERCENTAGE MULTIPILIER PattSpan (1,NPatt)
%:
      5.0   5.0

>IF NPatt>0, INTEGRAL SPAN PERCENTAGE MULTIPLIER PattISpan (1,NPatt) %/s:
      0.0

>IF NPatt>0, FILE NAME FOR EVERY PATTERN (UP TO 4 CHARACTERS PER NAME)
PattName (NPatt, 4):
      pat1
      pat2

>IF NPatt>0, INFLUENCE MATRIX FROM PATTERNS TO TARGETS Patt2Targ
(NCon,NPatt) (mm OR kN)/(mm OR kN):
      1.0   0.0
      0.0   1.0

>>>
>>>-------------------Other Influence Matrices----------------------
>>>

>INFLUENCE MATRIX FROM HEIDENHAIN TO TARGETS Heid2Targ (NCon,NCon) (mm OR
kN)/mm:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM TEMPOSONICS TO TARGETS Temp2Targ (NCon,NCon) (mm OR
kN)/mm:
```

```
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM LOAD CELLS TO TARGETS LCell2Targ (NCon,NCon) (mm OR
kN)/kN:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM FORCE2 CHANNEL TO TARGETS Force22Targ (NCon,NCon)
(mm OR kN)/unit:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM SPEED CHANNEL TO TARGETS Speed2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0   0.0
      0.0   0.0

>INFLUENCE MATRIX FROM LVDT CHANNEL TO TARGETS Lvdt2Targ (NCon,NCon) (mm
OR kN)/unit:
      0.0   0.0
      0.0   0.0

>>>
>>>----------------------PsD equation data---------------------------
>>>

>IF NDof>0, THEORETICAL MASS MATRIX Mass(NDof,NDof) kg:

>IF NDof>0, THEORETICAL ADDITIONAL STIFFNESS MATRIX StiffAdd(NDof,NDof)
N/m:

>IF NDof>0, THEORETICAL ADDITIONAL DAMPING MATRIX DampingAdd(NDof,NDof)
Ns/m:

>IF NDof>0, INITIAL DISPLACEMENT DisInit(1,NDof) m:

>IF NDof>0, INITIAL VELOCITY VelInit(1,NDof) m/s:

>>>To avoid restoring-force offset compensation introduce NFSampl=0

>IF NDof>0, NUMBER OF SAMPLINGS TO AVERAGE FOR RESTORING FORCE OFFSET
COMPUTATION NFSampl:

>IF NDof>0 AND NFSampl>0, PRESCRIBED RESTORING FORCE VALUE FOR OFFSET
COMPUTATION ResInit(1,NDof) N:

>IF NGacc>0, PROPORTIONAL SPAN PERCENTAGE MULTIPLIER GAccSpan(1,NGAcc) %:

>>>Accelerogram time increment must be equal to prototype time increment
>IF NGAcc>0, FILE NAME FOR EVERY GROUND ACCELEROGRAM (UP TO 4 CHARACTERS
PER NAME) GAccName(NGAcc,4):
```

```
>IF NDof>0 AND NGAcc>0, INFLUENCE MATRIX FROM GROUND MOTION TO DoF
GAcc2Dof(NDof,NGAcc) (m/s/s)/(m/s/s):

>IF NDof>0, INFLUENCE MATRIX FROM LOAD CELLS TO RESTORING FORCES
LCell2Res(NDof,NCon) N/kN:

>IF NDof>0, INFLUENCE MATRIX FROM DoF DISPLACEMENTS TO TARGETS
Dis2Targ(NCon,NDof) mm/m:

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO FSR
Force22FSR(NSR,NCon) kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO FSR Speed2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO FSR Lvdt2FSR(NSR,NCon)
kN/unit:

>IF NSR>0, INFLUENCE MATRIX FROM FORCE2 CHANNEL TO DSR
Force22DSR(NSR,NCon) mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM SPEED CHANNEL TO DSR Speed2DSR(NSR,NCon)
mm/unit:

>IF NSR>0, INFLUENCE MATRIX FROM LVDT CHANNEL TO DSR Lvdt2DSR(NSR,NCon)
mm/unit:


>>>Formula for strain-rate-compensation additional force at every device:
>>>FSRAdd = SRFacF0*FSR + SRFacD0*DSR + SRFacF1*FSRdot + SRFacD1*DSRdot


>IF NSR>0, FORCE CORRECTION FACTOR SRFacF0(1,NSR) kN/kN:

>IF NSR>0,  DISPLACEMENT CORRECTION FACTOR SRFacD0(1,NSR) kN/mm:

>IF NSR>0,  FORCE-DERIVATIVE CORRECTION FACTOR SRFacF1(1,NSR) kNs/kN:

>IF NSR>0,  DISPLACEMENT-DERIVATIVE CORRECTION FACTOR SRFacD1(1,NSR)
kNs/mm:

>IF NSR>0, INFLUENCE MATRIX FROM FSRAdd TO RESTORING FORCES
FSR2Res(NDof,NSR) N/kN:

>>>
>>>----------------------Algo Alarm Data-----------------------------
>>>

>ALGO_ALARM SUPERIOR LIMIT AT HEIDENHAIN HeidMax (1,NCon)
      2     2
>ALGO_ALARM INFERIOR LIMIT AT HEIDENHAIN HeidMin (1,NCon)
     -2    -2
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS TempMax (1,NCon)
      30    30
```

```
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS TempMin (1,NCon)
      -30   -30
>ALGO_ALARM SUPERIOR LIMIT AT TEMPOSONICS ABS TempAbsMax (1,NCon)
      1e10  1e10
>ALGO_ALARM INFERIOR LIMIT AT TEMPOSONICS ABS TempAbsMin (1,NCon)
      1e10  1e10
>ALGO_ALARM SUPERIOR LIMIT AT LOAD CELL FORCE LCellMax (1,NCon)
      40    40
>ALGO_ALARM INFERIOR LIMIT AT LOAD CELL FORCE LCellMin (1,NCon)
      -40   -40
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR ErrorMax (1,NCon)
      3     3
>ALGO_ALARM LIMIT AT ABSOLUTE ERROR AVERAGE ErrAvMax (1,NCon)
      1e10  1e10
>ALGO_ALARM LIMIT AT ABSOLUTE ENERGY ERROR AVERAGE EneErAvMax
      1e10
>ALGO_ALARM SUPERIOR LIMIT AT LVDT LvdtMax (1,NCon)
      1e10  1e10
>ALGO_ALARM INFERIOR LIMIT AT LVDT LvdtMin (1,NCon)
      -1e10 -1e10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION1 Press1Max (1,NCon)
      1e10  1e10
>ALGO_ALARM INFERIOR LIMIT AT PRESSION1 Press1Min (1,NCon)
      -1e10 -1e10
>ALGO_ALARM SUPERIOR LIMIT AT PRESSION2 Press2Max (1,NCon)
      1e10  1e10
>ALGO_ALARM INFERIOR LIMIT AT PRESSION2 Press2Min (1,NCon)
      -1e10 -1e10
>ALGO_ALARM SUPERIOR LIMIT AT SERVOVALVE ServoMax (1,NCon)
      5     5
>ALGO_ALARM INFERIOR LIMIT AT SERVOVALVE ServoMin (1,NCon)
      -5    -5
```

B. Pattern history pat1_pat.txt and pat2_pat.txt

```
>>>Pattern history
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line

>TITLE OF THE RECORD:
   up and down, 10mm
>NUMBER OF RECORD POINTS OF THIS HISTORY  NRecPatt:
10
>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS TimeRecIncr s:
      0.02
>RECORD VALUES Patt (NRecPatt,1) mm or kN:
0
0
10
```

```
10
0
-10
-10
0
0
0
```

```
>>>Pattern history
>>>PSDCYC03.DLL: JRC-ELSA general PsD and/or cyclic algorithm at one
master
>>>User comment lines are started with a #
# Example of user comment line

>TITLE OF THE RECORD:
   down and up, 10mm
>NUMBER OF RECORD POINTS OF THIS HISTORY  NRecPatt:
10
>PROTOTYPE TIME INCREMENT BETWEEN TWO RECORDS TimeRecIncr s:
      0.02
>RECORD VALUES Patt (NRecPatt,1) mm or kN:
0
0
-10
-10
0
10
10
0
0
0
```

INDEX

EXAMPLE M09: PsD for Snap Back m06 4DoF 4con

EXAMPLE e23: bench test 2acc 2con 2DoF

EXAMPLE e22: cyclic test 2con 2patterns

European Commission

EUR 23448 EN/2 – Joint Research Centre – Institute for the Protection and Security of the Citizen
Title: PSDCYC03.DLL User Manual
Author(s): Beatriz Zapico Blanco, F.Javier Molina
Luxembourg: Office for Official Publications of the European Communities
2008 – 83 pp. – 21 x 29.7 cm
EUR – Scientific and Technical Research series – ISSN 1018-5593
ISBN 978-92-79-09706-5
DOI 10.2788/92806

Abstract
A DLL (Dynamic Link Library) is a file of code containing functions that can be called from other executable code. The advantage of working with DLLs at ELSA PsD Master controller is in the modularity for programming the testing method and algorithm without the need to work with the master.exe program.
This new version of DLL allows the implementation of both PsD and Cyclic test, plus strain-rate effect compensation, re-start capabilities and a large variety of security alarms. In this manual the user will find a full explanation of how to use the DLL throughout some simple examples. In this Annex the user can find some other examples coming from real tests carried out at ELSA laboratory.

The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union. Close to the policy-making process, it serves the common interest of the Member States, while being independent of special interests, whether private or national.

LB-NB-23448-EN-C

ISBN 978-92-79-09706-5